Increasing Driver Flexibility through Personalized Menus and Incentives in Ridesharing and Crowdsourced Delivery Platforms

Hannah Horner[1a], Jennifer Pazour[2a,*], and John E. Mitchell[3a]

[a]Rensselaer Polytechnic Institute , 110 8th St, Troy, NY 12180, USA
[*]Corresponding Author
Declarations of interest: none

August 3, 2021

[1]horneh@rpi.edu, https://orcid.org/0000-0002-8921-6726
[2]pazouj@rpi.edu, https://orcid.org/0000-0002-0463-8744
[3]mitchj@rpi.edu, https://orcid.org/0000-0001-5087-4679

Abstract: Allowing drivers to choose which requests to fulfill provides drivers with much-needed autonomy in ridesharing and crowdsourced delivery platforms. While stochastic, a driver's acceptance of requests in their menu is influenced by the platform's offered compensation. Therefore, in this work, we create and solve an optimization model to determine personalized menus and incentives to offer drivers. We exploit variable properties to circumvent nonlinear variable relationships, formulating the model as a linear integer program. Stochastic driver responses are modeled as a sample of variable and fixed scenarios. An imposed premium counterbalances solution overfitting. Solution methods decompose and iterate, improving performance of computational experiments that use request/driver trip information from the Chicago Regional Transportation Network. Our approach outperforms alternative methods and a fixed incentives *menu-only* model by strategically using personalized incentives to prioritize promising matches and increase drivers' willingness to accept requests. This benefits both customers and drivers: the average driver income is increased by 4.1% compared to the menu-only model, and 96.6% of requests are matched (4.1% higher than the menu-only method). Higher incentives are offered when drivers are more likely to accept, while fewer incentives and menu slots are reserved for driver-request pairs less likely to be accepted.

1 Introduction

Ridesharing and crowdsourced delivery platforms have become common transportation sources, operating globally. In the United States, over a third of adults have used a ridesharing service (Jiang, 2019). The French platform BlaBlaCar has 40 million members worldwide, and 45% of smartphone owners in urban areas of Southeast Asia have used a ride-hailing platform (Mordor Intelligence, 2020). Similarly, scores of crowdsourced delivery platforms (e.g. Postmates, Instacart, Beelivery) are enabling retailers around the world to quickly deliver products to their customers. Currently 30% of retailers use some form of crowdsourced delivery, and almost 90% of retailers expect to use it by 2028 (Zebra, 2018). Customers submit a *request* for either a ride or a delivery, including an origin and destination. The request is then matched to a suitable *driver* that has identified themselves as available to drive for the platform. The customer pays the platform the specified price or *fare*, and the driver receives some wage or *compensation* from the platform for fulfilling the request.

The platform has a number of decisions to make in the design and implementation of its service, with critical ones being how to set pricing and compensation, and how to design the interaction structure, including matching decisions. Most common is for platforms to match drivers to requests, without providing autonomy of choice to either group. Exceptions exist in some platforms to offer customers multiple choices and corresponding prices; for example, Uber riders have the option to pool their ride with other riders for a discount or to pay extra to ride in a larger car (Uber, 2020). However, the driver side of the dispatching experience is usually quite constricted: most platforms propose a single request for a driver to accept or deny. In addition, rejecting too many requests can invoke penalties such as receiving fewer requests in the future (JC, 2019). Such lack of autonomy for drivers has led to drivers and labor activists in a number of states and countries pushing for ridesharing and delivery platforms to reclas-

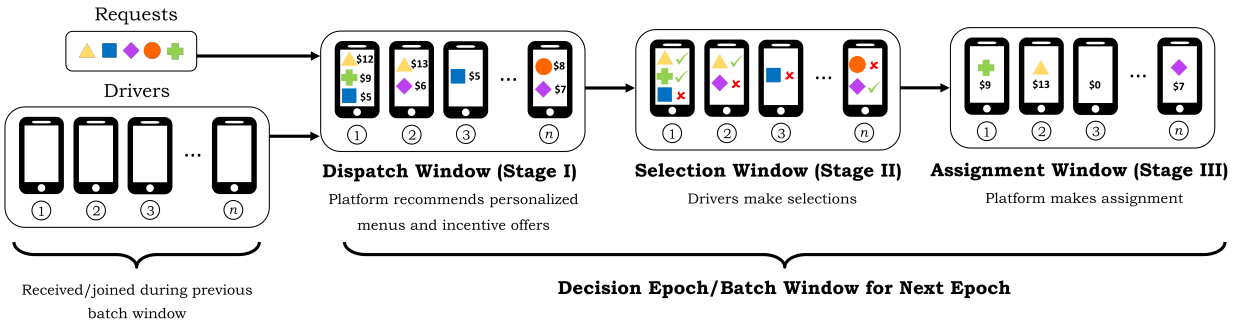


Figure 1: The three decision stages used to match current requests (colored shapes) and a set of n available drivers using the platform's app on their smartphones.

sify their drivers as employees, not independent contractors (Porterfield, 2021). The drivers' wages are typically calculated as a fixed fraction of the fare, perhaps with an applied surge multiplier reflecting the current supply-demand ratio. In this paper we propose a model to enhance the drivers' experience by creating small personalized menus of requests and corresponding compensation offers for each driver to choose from. The personalized compensation offers are initialized to be fare-based wages at a pay level similar to what current platforms offer, and the model is only allowed to increase this wage. The potential gain of this method is supported by the findings of Stiglic *et al.* (2015), which show that providing incentives to customers and/or drivers to make them slightly more flexible can significantly increase matching rates. In our case, we increase the flexibility of some drivers by increasing their compensation to make them more likely to accept a given request. There is also precedent and interest in the concept of more personalized incentives from practice, as Lyft has recently started offering drivers a wage bonus if they stay in or accept a request in one of the 'Personal Power Zones' shown to them based on their location (Lyft, 2021).

An overview of the process we model to match drivers and requests is shown in Figure 1. As is common in practice for platforms, we batch requests and drivers into discrete time windows or *batch windows*, which can be further divided into smaller batches based on geographical region (Qin *et al.*, 2020). The matching process, taking place over a *decision epoch*, is optimized over the set of drivers and requests in the same batch. There are three decision stages that make up each decision epoch: first, Stage I is the *dispatch window*, where the platform creates personalized menus of requests and incentives to send to the drivers. The drivers then individually decide which requests (if any) to accept from their menu during the *selection window* (Stage II), and in the *assignment window* (Stage III), the platform uses the set of driver responses to find the optimal feasible assignment where each driver is matched to at most one request. After the matches are made, the drivers and requests that joined the platform during the epoch are batched along with any unmatched drivers/requests that decide to stay and attempt to be matched again, and the next decision epoch begins.

The final results of the assignment in Stage III influence the optimal platform decisions in Stage I, so a model encompassing all three stages as a single optimization program is desirable. The platform may compile driver data from user history, but the platform will not have perfect knowledge of the drivers'

selections before they make them. We therefore model the drivers' behavior stochastically by estimating each driver's willingness to accept each request and assume that the offered compensation affects this probabilistic value. A prevalent challenge in practice is dealing with drivers rejecting their assignment, as this forces both the customer and the driver to wait until the next decision epoch before they can reattempt to be matched. Rejection is often curbed by threat of penalty to drivers, but in places where drivers are not penalized for rejections, drivers are rejecting their assignment more often (Marshall, 2020). The platform can mitigate uncertainty in driver behavior by recommending multiple requests to each driver, and personalizing the incentives further improves the quality of assignments, as the platform can strategically offer additional compensation to boost the successful matching rate. This can be helpful to encourage drivers who otherwise would not have desirable menu options to accept a request and participate in the system. The platform can also increase the incentives for less-popular requests to increase the probability of them being successfully matched. Creating both menus and incentives for every driver is a complex and challenging undertaking; decisions must be made out of a combinatorially vast solution space, the optimal menus are interdependent with the optimal incentives, and these decisions must also consider how their influence is propagated through the stochastic driver selections to affect other drivers and the platform's recourse options.

Ample research finds optimal compensation (and/or pricing), but a request's compensation level is almost always required to be the same across all drivers, and often across all requests (e.g. determining surge price multipliers). Few matching models offer multiple options to drivers or consider stochastic driver behavior; instead, matches meeting some requirement (known a priori) are assumed feasible. Thus, our objective is to develop a method able to provide insights on how a platform can match drivers and requests using personalized menus and incentives. The contributions of our paper are as follows. We are the first to create both optimized personal menus of requests and optimized personal compensation offers for each request in each menu. Drivers' stochastic and endogenous selection behavior creates nonlinear relationships among assignment and compensation decision variables; by exploiting structural properties of the personalized compensation decisions, we formulate the problem as a stochastic linear integer program. This linear model finds both menus and corresponding incentives using a method inspired by the Sample Average Approximation (SAA) in that it samples different scenarios of driver behavior (i.e. a set of all driver's responses to all requests). For SAA, these scenarios are generated as inputs to the model; in contrast, we must model the scenarios as variables because our decision variables (compensation) affect driver behavior. To counterbalance the solver's ability to exploit the variability of driver decisions (resulting in overfitting), we impose a premium that requires the solver to offer higher wages than is needed for the solution's driver participation levels. We then develop a series of solution methods to improve performance. We decompose the formulation into two separate problems of creating menus for given compensation values (using the formulation from Horner *et al.* (2021)) and deciding incentive offers for fixed menus (by solving a modified integer program). To reduce overfitting, we include fixed scenarios of different driver selections in our model in addition to the variable scenarios. Lastly, we iterate between the two subproblems and use a time-efficient performance test to select which iteration's solution to return. Computational results—using ridesharing in Chicago as a case

study—demonstrate the effectiveness of these solution methods and provide user setting recommendations. Additional experiments establish our model's merit in comparison with alternative methods. We analyze solution decision variables to gain insights into how a platform can match requests and drivers using menus and additional incentives.

The remainder of this paper is structured as follows. Section 2 reviews the current ridesharing and crowdsourced delivery literature in the context of our problem. In Section 3 we develop our model and introduce solution methods along with our method for performance analysis. We present our computational experiments and results in Section 4 and discuss our conclusions in Section 5.

2 Literature Review

Peer-to-peer transportation systems have become a popular research topic in recent years, with a wide body of literature covering a variety of subtopics and decisions. Overviews of existing ridesharing literature is presented in Furuhata *et al.* (2013), Tafreshian *et al.* (2020), and Wang & Yang (2019), and crowdsourced delivery models are reviewed in Alnaggar *et al.* (2019) and Le *et al.* (2019). This work optimizes both individualized menus to send to drivers and the individualized incentives for each request in each menu. Thus, we examine existing ridesharing and crowdsourced delivery literature that incorporates request pricing and/or driver compensation decisions, and order dispatching (also known as matching) literature that creates driver menus or offers some other alternative to enhance the drivers' autonomy working with a platform.

Our review of pricing and compensation literature consists of first describing the influence price and compensation has on customers and drivers, followed by a discussion of methods and decision factors used to determine price and compensation. Many papers investigate pricing and/or compensation for a ridesharing platform, as both of these decisions affect a platform's bottom line and therefore its viability. Models that have price/compensation as a decision variable have some level of endogeny, our model included. Our formulation makes driver compensation decisions jointly with menu decisions and the compensation, which affect the resulting decisions made by each driver to accept or reject each request. For other papers, the effect of pricing/compensation can manifest in different ways, such as drivers deciding to start or end a session with the platform based on compensation level (Nourinejad & Ramezani, 2019). The price/compensation can also be used along with other factors such as wait time or detour to calculate a more comprehensive participation cost or utility for the riders/drivers, which must be at a certain level for the rider/driver to be matched (Rasulkhani & Chow, 2019; Sun *et al.*, 2019). For equilibrium models, this cost must not only meet a threshold, it must be at least as good as all alternative options for the riders and drivers in order to achieve equilibrium (Liu & Li, 2017; Ma *et al.*, 2020a). The price can also affect the general demand (request) level, which in turn affects the supply (driver) level (Cachon *et al.*, 2017; Luo & Saigal, 2017). Though the effects of pricing and compensation can be nuanced, the end result is that they determine the population of drivers and customers that will participate, and/or which drivers and requests can be matched to each other.

A variety of methods are used to make pricing and compensation decisions. Nourinejad & Ramezani

(2019) solve for the optimal pricing and compensation for each time period using a receding horizon approach implemented with a model predictive control (MPC) framework. Chen *et al.* (2018a) find the optimal pricing scheme by capturing driver populations at different locations as states in a Markov Decision Process. A popular approach is to create an economic model and find the equilibrium space to decide pricing (as pricing affects the equilibrium) (Bimpikis *et al.*, 2019; Castillo *et al.*, 2018; Ma *et al.*, 2020a). Many of these models capture the matching process as a queuing model, where drivers and riders must wait in a queue when the departure rate exceeds the system capacity (Bai *et al.*, 2019; Banerjee *et al.*, 2015; Liu & Li, 2017; Yan *et al.*, 2020). These approaches tend to model drivers and riders as a generalized population that may become more specific in numerical experiments. In contrast, because our formulation creates personalized menus and prices simultaneously for many drivers, the personal parameters for each driver are explicitly captured in our model.

A number of different factors can be considered for deciding price(s). Some depend on the current supply and demand levels in the platform (Bai *et al.*, 2019; Cachon *et al.*, 2017) or future supply and demand (Asghari & Shahabi, 2018), which sometimes manifests as a surge multiplier applied to a base fare (Castillo *et al.*, 2018; Zha *et al.*, 2018). The supply and demand can also consider location, and the price is determined by the supply and demand at the request's origin and destination (Asghari & Shahabi, 2018; Ma *et al.*, 2020a). Ke *et al.* (2020) and de Ruijter *et al.* (2020) investigate the fare discounts that are possible in a ride-pooling market.

Research regarding demand pricing is abundant; conversely, driver compensation is often not explored as an independent decision variable. Usually the driver's compensation is a fixed percent of the fare (Asghari & Shahabi, 2018; Castillo *et al.*, 2018; Zha *et al.*, 2018), so it is affected linearly by the pricing decision variable. Some models allow this commission percent to vary based on supply, demand, or another factor (Bai *et al.*, 2019; Cachon *et al.*, 2017; Chen *et al.*, 2020; Hu & Zhou, 2019). For these papers, the prices and wages are the same flat amount or the same wage per distance for all riders/drivers. Barbosa (2019) and Hong *et al.* (2019) both model delivery platforms with nonvariable prices that focus on the optimal wage to offer to drivers. Barbosa (2019) determines the optimal subset of requests (with remaining requests fulfilled by a dedicated fleet) and the wage offer to propose to all available drivers, who have a price-dependent probability of rejecting each request. Hong *et al.* (2019) determines the route and per-distance reward for a given request to broadcast to drivers who can then make offers to fill some part of the route. Le *et al.* (2021) allow for both personalized pricing and compensation in a crowdsourced delivery platform, but customers/drivers are assumed to accept any price/wage named by the platform as long as it is below/above their willingness to pay/expected earnings which are known and fixed. Therefore allowing both price and wage to vary results in a simplified solution where prices/wages are at their upper/lower bounds. To our knowledge our paper is the only model that isolates compensation decisions given fixed pricing, and simultaneously decides personalized menus and incentives to match drivers and requests.

We create menus with multiple options to improve driver autonomy. Many ridesharing and crowdsourced delivery approaches assume drivers are willing to accept any request(s) assigned to them (Lei *et al.*, 2019; Wang *et al.*, 2016). Other models guarantee that the drivers' extra driving time/distance does

not exceed an upper bound (Masoud & Jayakrishnan, 2017; Stiglic *et al.*, 2015), or that the route of their matched trip will allow them to depart from their origin and arrive at their destination within their desired time window (Chen *et al.*, 2018b; Najmi *et al.*, 2017), but as long as these constraints are met, drivers are assumed compliant. Equilibrium models also tend to consider some level of driver preferences, as drivers choose to fulfill fewer or no requests if their utility/wage for their match is not sufficiently high (Liu & Li, 2017; Ma *et al.*, 2020b). Auction formats where drivers accept or bid on broadcasted requests allow for high driver autonomy, but also require a lot of participation from the driver, as they have to decide whether or not to bid on every request (Barbosa, 2019; Hong *et al.*, 2019; Sun *et al.*, 2019).

A handful of approaches create menus for riders, where differences between menu items can be the mode of transportation (taxi, rideshare, and ride pool with multiple riders) (Wu *et al.*, 2018), the route and resulting arrival time (Lei *et al.*, 2019), etc. Each option has a corresponding price that also influences the riders' selections. Jacob & Roet-Green (2018) and Lei *et al.* (2019) simultaneously optimize the menu content/type and the prices to offer, though for Jacob & Roet-Green (2018) the prices and menu options (solo ride, pool ride, or both) are the same for all riders. These types of rider menus are fundamentally different from our driver menus, as the riders know what price they will pay for their ride as soon as they make a selection. Conversely, in our formulation, the drivers do not know what their income will be until all drivers have made their selections, as matches are made based on the complete set of driver decisions. Li *et al.* (2019b) does create rider menus with dynamic pricing that updates the ride-pooling discount as additional riders may be added to the rider's car during the trip. However these prices follow a deterministic formula and are not optimized.

Only a few papers offer menus to drivers. Using menus requires some form of driver feedback; for our approach, each driver's decision to accept or reject each request in their menu is the result of a stochastic process that depends on compensation and effort required. Existing models use a similar mechanic, but instead of receiving a menu, drivers make decisions on either a single request or all requests (Barbosa, 2019; Di Febbraro *et al.*, 2013; Gdowska *et al.*, 2018), and thus the combinatorial optimization problem of menu creation is not addressed. Li *et al.* (2019a) recommends all requests with an origin within a certain radius of the driver's location, and drivers make selections to maximize their total discounted reward. Mofidi & Pazour (2019) uses a hierarchical model that creates personalized menus with fixed compensations for drivers, whose responses are assumed known in advance by the platform as driver utilities are known and deterministic. Because driver selections are deterministic, there exists an optimal menu set in which each driver receives at most one request, and therefore they rely on simulation to recommend driver menus. Ausseil *et al.* (2021) use a multiple scenario approach to determine fixed-sized and fixed-compensation menus for dynamically arriving drivers and riders. Horner *et al.* (2021) use a hierarchical method to optimize personalized menus for drivers under fixed compensation, where a driver's probability of accepting a given request is determined by their wage and extra driving time. We expand upon the literature providing menus to drivers by allowing the platform to also decide the extra compensation to offer to each driver for each request in their menu, which then affects the drivers' selection decisions. To our knowledge we are the first to present a formulation that both creates optimized menus for drivers and optimized individual incentives for each request in each driver's menu.

3 Methods

The problem of creating personalized menus and incentives for drivers is complex due to the drivers' stochastic and endogenous selection behavior, which creates nonlinear relationships between the assignment and compensation decision variables. Further, the hierarchical natures and vast decision space and driver selection space create computational challenges. Thus, in Section 3.1, we exploit structural properties to formulate the problem as a stochastic linear integer program. We present multiple specialized solution methods for the formulated problem in Section 3.2. In Section 3.3 we describe the process for obtaining an unbiased performance estimate of solutions, and in Section 3.4 we summarize the different types of scenarios we use in our formulation, in solution methods, etc.

3.1 Initial Formulation

We formulate a mathematical program to determine the personalized menu of requests and associated request incentives a platform should send to each currently available driver. Table 1 displays the notation used in our formulation, solution methods, and performance analysis. The request's trip-dependent fares paid by the customers are assumed to remain constant, but which drivers are offered which requests and how each fare is split between the platform and a given driver are decision variables. This mathematical program represents a three-stage process where in Stage I, the platform determines the menus ($\boldsymbol{x}$ variables) and compensations ($\boldsymbol{comp}$ variables) to send to the currently available drivers. Then, in Stage II the drivers send feedback ($\boldsymbol{y}$ variables) on which request(s) they are willing to fulfill from their menus (if any). In Stage III, the platform sends out a valid assignment ($\boldsymbol{v}$ variables) of requests to drivers based on received driver feedback. This process of matching drivers and requests takes place once for each epoch. After the drivers are matched to requests, the set of available drivers (M) and the set of unmatched requests (N) is updated, batching together any drivers and requests that have joined the platform within the last epoch and any previously unmatched customers/drivers who choose to stay for an additional epoch in attempt to be matched. The platform then reoptimizes the three-stage process for the new epoch. In this paper we consider a single decision epoch, focusing on one round of sending menus to drivers and creating an assignment based on their responses.

The formulation aims to maximize the platform's expected objective, which can be as simple as revenue minus wages paid to drivers, or can include additional factors such as a bonus for each assignment made to prioritize match rate, or to prioritize efficient matches, etc. To driver j, the net utility for fulfilling request i is denoted as $util_{ij}$, which is driver j's utility for fulfilling request i minus the utility they receive from not fulfilling any requests. While $util_{ij}$ is influenced by the platform's offered compensation for request i, we assume $util_{ij}$ does not change based on which other requests are offered in a driver's own menu or the compensation offers for these requests. Further, each driver interacts with the platform independently of other drivers, thus, $util_{ij}$ is not influenced by any other drivers' utilities, menus, or compensations. Each driver is willing to accept all requests in their menu with a positive $util_{ij}$, and will accept zero requests if all of their menu's requests have a non-positive $util_{ij}$. A main challenge with this formulation is that the platform does not have perfect knowledge of these net utility

Table 1: Overview of sets, indices, parameters, and variables.

Notation	Description
Sets	
M	Set of all requests, $M = 1, 2, ..., m$
N	Set of all available drivers, $N = 1, 2, ..., n$
$\Omega/\bar{\Omega}/\hat{\Omega}/\dot{\Omega}$	Set of sampled variable/SLSF/fixed/test scenarios
Indices	
i	Index of request, $i \in M$
j	Index of driver, $j \in N$
$s/\bar{s}/\hat{s}/\dot{s}$	Index of variable/SLSF/fixed/test scenario, $s \in \Omega/\bar{s} \in \bar{\Omega}/\hat{s} \in \hat{\Omega}/\dot{s} \in \dot{\Omega}$
Parameters	
m, n	Number of requests and drivers, respectively
$\hat{c}_{ij}$	Platform's reward, not counting wages, for driver j fulfilling request i
$match_i$	Bonus received for successfully matching request i
d_{ij}	Penalty for each SLSF/test scen. that driv. j accepts req. i and is an unhap. driv.
ℓ, θ	Minimum and maximum menu size, respectively
$util_{ij}$	Net utility driver j receives from fulfilling request i
p_{ij}	Probability that driver j is willing to accept request i
$\bar{y}^{\bar{s}}_{ij}$	Willingness: is 1 if driv. j is willing to accept req. i in scen. $\bar{s}$; otherwise is 0
$\hat{y}^{\hat{s}}_{ij}/\dot{y}^{\dot{s}}_{ij}$	Selections: is 1 if driv. j accepts req. i in scen. $\hat{s}/\dot{s}$; otherwise is 0
$prem$	Premium on conversion factor from compensation to participation
α_{ij}	Upper bound of t_{ij}; maximum compensation amount such that $p_{ij} = 0$
β_{ij}	Upper bound of u_{ij}
$wage_{min}$	Maximum wage per unit of effort such that $p_{ij} = 0$
$wage_{max}$	Minimum wage per unit of effort such that $p_{ij} = 1$
$mincomp_i$	Minimum offer allowed as compensation for request i
$fare_i$	Fare paid for request i to the platform
$effort_{ij}$	Effort required for driver j to fulfill request i
$miles_i$	Travel distance in miles of request i's trip
$mins_i$	Travel time in minutes of request i's trip
$mins_{ij}$	Travel time in mins from driv. j to req. i (req. i's wait time if matched with driv. j)
Variables	
x_{ij}	Menu: $x_{ij} = 1$ if request i is recommended to driver j; otherwise $x_{ij} = 0$
$comp_{ij}$	Amount paid to driver j if driver j fulfills request i
t_{ij}, u_{ij}	The two components of compensation, $comp_{ij} = t_{ij} + u_{ij}$
$\ddot{p}_{ij}$	Variable approximation of p_{ij}
y^s_{ij}	Driv. selection: is 1 if $x_{ij} = 1$ and driv. j accepts req. i in scen. s; otherwise $y^s_{ij} = 0$
$v^s_{ij}/\bar{v}^{\bar{s}}_{ij}/\hat{v}^{\hat{s}}_{ij}/\dot{v}^{\dot{s}}_{ij}$	Assignment: is 1 if req. i is assigned to driv. j in scen. $s/\bar{s}/\hat{s}/\dot{s}$; otherwise is 0
$\phi^s_{ij}/\hat{\phi}^{\hat{s}}_{ij}$	Represents variable product $u_{ij}\phi_{ij}/u_{ij}\hat{\phi}^{\hat{s}}_{ij}$
$\bar{z}^{\bar{s}}_{ij}/\dot{z}^{\dot{s}}_{ij}$	Unhappy Driver: is 1 if $\bar{y}^{\bar{s}}_{ij}/\dot{y}^{\dot{s}}_{ij}$ is 1 but driver j receives no assignment in scenario $\bar{s}/\dot{s}$; otherwise is 0

values. To address the stochasticity of driver behavior, we use a method inspired by the Sample Average Approximation (SAA) method, where driver responses are approximated by using a small sample Ω of training scenarios, and the expected value of the objective function is optimized across these scenarios

(Kleywegt *et al.*, 2002). For the standard SAA method, the scenarios are fixed and generated as inputs to the program. However, for our problem, driver behavior is endogenous as it is affected by the compensation offers, which are decision variables, so these scenarios capturing driver selections are instead variable. The Stage II driver feedback in each variable training scenario $s \in \Omega$ is represented by the binary $\boldsymbol{y}$ variables. To capture sufficient information to know exactly what any driver will accept from their menu in scenario s, we define a variable scenario s to be a set of y^s_{ij} values for every (i, j) pair such that $i \in M$, $j \in N$, and $x_{ij} = 1$ (we constrain y^s_{ij} to be zero for all $i \in M$, $j \in N$ such that $x_{ij} = 0$).

Developing a joint menu and compensation model is challenging because both compensation and menu variable values affect the outcome of the drivers' selections. In each scenario, driver j is either willing to accept request i if it is in their menu or they will reject it, and so on for every $j \in M$ and $i \in N$. The probability driver j is willing to accept request i if offered in their menu, i.e. $\mathbb{P}(util_{ij} > 0)$, is denoted as p_{ij}. Because compensation is a decision variable, so is p_{ij}. We model the connection between driver j's willingness p_{ij}, compensation $comp_{ij}$ for request i, and the effort $effort_{ij}$ required to fulfill the request, in Equation 1. Specifically, we model p_{ij} as a piecewise linear continuous function of $comp_{ij}/effort_{ij}$, i.e. driver j's effective wage per unit of effort for fulfilling request i.

$$p_{ij} = \begin{cases} 0, & \text{if } \frac{comp_{ij}}{effort_{ij}} < wage_{min} \quad (1a) \\ \frac{\frac{comp_{ij}}{effort_{ij}} - wage_{min}}{wage_{max} - wage_{min}} & \text{if } wage_{min} \le \frac{comp_{ij}}{effort_{ij}} < wage_{max} \quad (1b) \\ 1 & \text{if } wage_{max} \le \frac{comp_{ij}}{effort_{ij}} \quad (1c) \end{cases}$$

To capture the relationship between $comp_{ij}$ and p_{ij} as a set of mathematical constraints, we split $comp_{ij}$ into two components, parameter α_{ij} and variable u_{ij}, such that $0 \le u_{ij} \le \beta_{ij}$ and $comp_{ij} = \alpha_{ij} + u_{ij}$. We can model p_{ij} exactly via a linear constraint relating p_{ij} with u_{ij} and α_{ij}, when α_{ij} is defined as the highest compensation a driver can receive such that $p_{ij} = 0$. This is the lower break point in the piecewise p_{ij} formula, so $\alpha_{ij} = wage_{min} \cdot effort_{ij}$. By our definition, $comp_{ij}$ cannot be lower than α_{ij}, which can be quite high if a driver-request pair requires a lot of effort. Allowing α_{ij} to be a variable would enable lower compensations, but would require a complementarity constraint between u_{ij} and α_{ij} for p_{ij} to be accurately represented by the linear constraint. Instead of introducing this source of nonlinearity to the model, we leave α_{ij} as a parameter because allowing the offer to be below $wage_{min} \cdot effort_{ij}$ does not benefit the formulation. This is because if $comp_{ij} < wage_{min} \cdot effort_{ij}$, then by (1) we know that there is zero probability of driver j accepting request i, meaning there is a zero probability of request i being assigned to driver j.

The upper bound for u_{ij}, β_{ij}, is set by the platform, for example, to be the fare for request i. Using the decomposition $comp_{ij} = \alpha_{ij} + u_{ij}$, we are able to express p_{ij} as the linear formula of variables $p_{ij} = \left(wage_{min}/(\alpha_{ij}(wage_{max} - wage_{min}))\right) \cdot u_{ij}$. This formula does not accurately capture (1c) i.e p_{ij} values when $comp_{ij} \ge wage_{max} \cdot effort_{ij}$, as p_{ij} represents a probability and should not be higher than 1. Enforcing p_{ij} to not exceed one would require additional constraints and place unnecessary and potentially detrimental restrictions on the compensation. For this reason, we use $\ddot{\boldsymbol{p}}$ to represent

this driver willingness variable in our formulations instead of $\boldsymbol{p}$, where $\ddot{\boldsymbol{p}}$ follows the formula (1b) for all $comp_{ij}/effort_{ij} \geq wage_{min}$. We show later in this section in our discussion of imposed premium that allowing $\ddot{\boldsymbol{p}}$ to exceed one does not hinder our model.

With the relationship between compensation and driver behavior established, we need to make constraints on the y^s_{ij} so that they collectively resemble random possible selection scenarios. For each variable training scenario $s \in \Omega$, each driver $j \in N$ either accepts each request $i \in M$ or doesn't. Because the driver selection outcomes are uncertain and the result of a random Bernoulli process that depends on p_{ij} values, the values of the y^s_{ij} are not deterministic. With this in mind, we enforce the relationship not on specific outcomes, but instead enforce average performance of a set of scenarios. That is, on average the y^s_{ij} follow an independent Bernoulli random distribution for each driver-request pair in a menu, so for a sample Ω of variable scenarios, on average $p_{ij} \cdot |\Omega|$ scenarios should have $y^s_{ij} = 1$ for each $i \in M$ and $j \in N$. The use of average performance constraints for y^s_{ij} means that the solver may produce a solution that meets the required ratio of y^s_{ij} values but strategically allocates driver participation so that good recourse assignments are possible in all variable training scenarios, but not all possible scenarios (i.e., our model is overfitting).

To counterbalance overfitting due to strategic placement of positive y^s_{ij} values, we introduce the concept of a *premium* that forces the optimization program to offer higher compensation rates than what is required for the driver participation in the approximated optimal solution. If the compensation is higher than the optimization solution plans for, a scenario randomly generated from the p_{ij} values under the solution's compensations (i.e. a *test scenario* used to judge the performance of the model under the training scenarios) will likely have higher driver participation that increases the chance of having good recourse assignments. We use a premium multiplier $prem$, where $prem \in (0,1]$, to influence the participation ratio, yielding the constraint (9). This constraint has $prem \cdot \ddot{p}_{ij}$ as the upper bound for $(1/|\Omega|) \cdot \sum_{s \in \Omega} y^s_{ij}$, as the purpose of $\ddot{p}_{ij}$ is to dictate the level of willingness driver j has to accept request i. A premium value of one means there is no change to the participation ratio, and a lower premium multiplier indicates a steeper compensation premium paid for driver participation. While the premium can technically have any value in $(0,1]$, having a low premium value, for example below 0.5, will hinder the model: it becomes too heavily constrained to find good solutions, as drivers require enough compensation to be more than twice as willing to accept a request than is needed in the solution. Because we add this premium and have an inequality constraint instead of an equality, it is advantageous to allow the $\ddot{p}_{ij}$ variable value to exceed 1 in the model, as mentioned in the discussion of the $\ddot{p}_{ij}$ value constraints. This is because, for example, having driver j accept request i in 100% of variable training scenarios requires the compensation to yield $\ddot{p}_{ij} = 1/prem$ which is higher than one if $prem < 1$.

The variables and constraints described thus far along with a few simple menu and assignment constraints yield our linear compensation formulation (LCF) given by (2)-(15). The objective function (2) maximizes the expected value of the platform's total net utility for the recourse assignments across the variable training scenarios. This net utility is calculated and summed for each driver j, request i, and variable training scenario s, and the net utility is zero if the driver-request pair is not assigned in training scenario s. The reward for the platform if driver j fulfills request i, including the fare collected and any

other factors included in the objective function (e.g. matching bonus), is constant for any compensation offered to driver j, so the reward, not including the wages paid, is consolidated into a single parameter $\hat{c}_{ij}$. The net utility for assigning request i to driver j in training scenario s is therefore $\hat{c}_{ij}$ minus the wage paid to driver j, $\alpha_{ij} + u_{ij}$. We can capture the net utility received for a given request i, driver j, and scenario s by multiplying $\hat{c}_{ij} - u_{ij} - \alpha_{ij}$ by the assignment v_{ij}^s, so the platform only receives the utility if the driver-request pair is assigned in that scenario. This does create a variable product term, $u_{ij}v_{ij}^s$; we note that v_{ij}^s is binary, so we can simply create a new variable ϕ_{ij}^s to represent $u_{ij}v_{ij}^s$ and add McCormick inequalities to ensure ϕ_{ij}^s takes the correct value. Each variable training scenario has a different likelihood of occurring which would normally be used as a weight in calculating the objective's expected value. However, this likelihood is a function of the $\ddot{p}_{ij}$ and y_{ij}^s values, which would yield a product of many variables term in the objective. To avoid adding an intractable level of complexity in the model, we weight each variable training scenario equally.

$$\max_{\substack{\ddot{p},u,v,\\ x,y,\phi}} \quad \frac{1}{|\Omega|}\sum_{s\in\Omega}\sum_{i\in M}\sum_{j\in N}\left((\hat{c}_{ij}-\alpha_{ij})v_{ij}^s - \phi_{ij}^s\right) \tag{2}$$

$$\text{s.t.} \quad \ell \le \sum_{i\in M} x_{ij} \le \theta \qquad \forall j \in N \tag{3}$$

$$y_{ij}^s \le x_{ij} \qquad \forall s\in\Omega, \quad \forall i \in M, \quad \forall j \in N \tag{4}$$

$$v_{ij}^s \le y_{ij}^s \qquad \forall s\in\Omega, \quad \forall i \in M, \quad \forall j \in N \tag{5}$$

$$\sum_{i\in M} v_{ij}^s \le 1 \qquad \forall s\in\Omega, \quad \forall j \in N \tag{6}$$

$$\sum_{j\in N} v_{ij}^s \le 1 \qquad \forall s\in\Omega, \quad \forall i \in M \tag{7}$$

$$\ddot{p}_{ij} = \frac{wage_{min}}{\alpha_{ij}(wage_{max} - wage_{min})}u_{ij} \qquad \forall i \in M, \quad \forall j \in N \tag{8}$$

$$\frac{1}{|\Omega|}\sum_{s\in\Omega} y_{ij}^s \le prem\cdot\ddot{p}_{ij} \qquad \forall i \in M, \quad \forall j \in N \tag{9}$$

$$\alpha_{ij} + u_{ij} \ge mincomp_i \cdot x_{ij} \qquad \forall i \in M, \quad \forall j \in N \tag{10}$$

$$0 \le u_{ij} \le \beta_{ij}x_{ij} \qquad \forall i \in M, \quad \forall j \in N \tag{11}$$

$$0 \le \phi_{ij}^s \le u_{ij} \qquad \forall s\in\Omega, \quad \forall i \in M, \quad \forall j \in N \tag{12}$$

$$\beta_{ij}v_{ij}^s + u_{ij} - \beta_{ij} \le \phi_{ij}^s \le \beta_{ij}v_{ij}^s \qquad \forall s\in\Omega, \quad \forall i \in M, \quad \forall j \in N \tag{13}$$

$$x_{ij} \le \sum_{s\in\Omega} v_{ij}^s \qquad \forall i \in M, \quad \forall j \in N \tag{14}$$

$$v_{ij}^s, x_{ij}, y_{ij}^s \in \{0,1\} \qquad \forall s\in\Omega, \quad \forall i \in M, \quad \forall j \in N \tag{15}$$

Constraints (3)-(7) ensure a valid menu and recourse assignment: the menu size for each driver must fall within a pre-specified range, $[l,\theta]$ (constraint (3)), driver j cannot accept request i in training scenario s unless request i appears in their menu (constraint (4)), and request i can only be assigned to driver j in training scenario s if driver j accepted request i in training scenario s (constraint (5)). Each driver can only be assigned one request in each scenario (constraint (6)) and each request can be as-

signed to at most one driver per scenario (constraint (7)). Constraint (8) implements the $\ddot{p}_{ij}$ formula according to the compensation $comp_{ij} = \alpha_{ij} + u_{ij}$ and Equation 1 (except that $\ddot{p}_{ij}$ values for all compensations above α_{ij} follow (1b) instead of capping at $\ddot{p}_{ij} = 1$). Constraint (9) ensures that drivers don't have a higher average willingness to accept each request than the calculated $\ddot{p}_{ij}$ would suggest, with the premium $prem$ included to reduce overfitting. Constraint (10) is added to improve driver compensation, by requiring each offer for request i (if i appears in the given driver's menu) to be at least some platform-specified dollar amount $mincomp_i$, which can be a flat amount, a function of the fare collected, etc. Constraint (11) enforces the bounds of u_{ij}, and forces u_{ij} to be zero for driver-request pairs not in the menu set. This is to make a tighter formulation that also allows for easier processing of the solution as there are no trivial wage offers above α_{ij}. McCormick Constraints (12) and (13) ensure that $\phi^s_{ij} = u_{ij} v^s_{ij}$. To increase tightness and ease solution processing, constraint (14) requires all items in the menu set to be assigned to the corresponding driver(s) in at least one scenario. Lastly, (15) constrains $\boldsymbol{v}$, $\boldsymbol{x}$, and $\boldsymbol{y}$ to be binary so there are no partial assignments, driver selections, or menu items.

3.2 Solution Methods

In this section we present a series of solution methods, which culminate in our iterative method SOL-IT. An outline of the process for each method is shown in Figure 2, and an algorithmic overview of SOL-IT is presented in Algorithm A.1 in Appendix A. These solution methods require four different types of scenarios, therefore, in Section 3.4, we summarize their differences. The term *variable LCF scenarios* is used interchangeably with variable training scenarios.

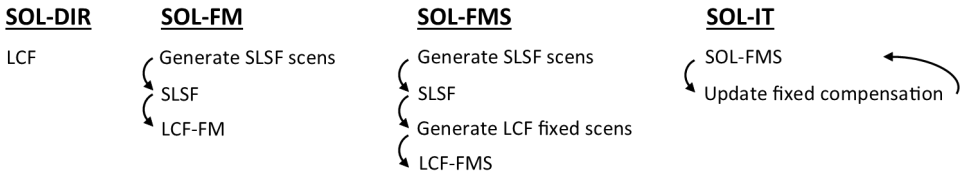


Figure 2: Formulation outline for the different solution methods.

3.2.1 Direct Method, SOL-DIR

LCF is a linear integer program that can simultaneously produce personalized menus and compensations to drivers. We refer to the solution method of solving LCF directly, using off-the-shelf solvers for a given computational budget, as SOL-DIR. However, as we show in our computational results, SOL-DIR solutions are not able to achieve comparable solutions—in terms of objective value and the expected number of matches—to solutions produced using a method to find menus under a fixed compensation scheme. Therefore, we design specialized solution methods to create driver menus and compensation.

3.2.2 SOL-FM, the Solution Method with Fixed Menus

Our second method decomposes LCF into two separate decision tasks, namely determining a promising menu set given fixed compensation, and then determining optimal compensation offers for that menu set. We refer to this two-step method as SOL-FM, the Solution Method with Fixed Menus.

To find the menus, we employ the Single-Level Stochastic Formulation (SLSF) method presented in Horner *et al.* (2021) and displayed in Appendix B. This formulation finds menus under fixed compensation by implementing the SAA method with fixed driver behavior training scenarios or *SLSF training scenarios* generated as inputs, since compensation and therefore p_{ij} is fixed. We denote an SLSF training scenario as $\bar{s}$, and the sample set of SLSF training scenarios as $\bar{\Omega}$. Because the menus are not fixed at this stage, we must generate driver selections for every driver-request pair, making them slightly different from the variable training scenarios. These selections represent the driver's response *if* the given request is offered in their menu, as their willingness to accept a request is not affected by whether or not it is actually offered to them. This information is recorded in $\bar{\boldsymbol{y}}^{\bar{s}}$ variables, where $\bar{y}^{\bar{s}}_{ij} = 1$ if driver j is willing to accept request i in scenario $\bar{s}$ and equals 0 if not.

To find the optimal compensations, given the menu set $\boldsymbol{x}^*$ returned from SLSF, we use a modified version of LCF where $\boldsymbol{x}$ is fixed to be $\boldsymbol{x}^*$, so constraint (3) is removed. We also remove constraint (14), as its purpose is to filter out trivial menu items which is not necessary when the menu is fixed. Therefore the compensation optimization formulation with fixed menus, referred to as LCF-FM, is given by (2),(4)-(13), and (15), optimized over $\ddot{\boldsymbol{p}}$, $\boldsymbol{u}$, $\boldsymbol{v}$, $\boldsymbol{y}$, and $\boldsymbol{\phi}$.

3.2.3 SOL-FMS, Fixed Menus and Scenarios

Our third solution method continues to use a two-step process of finding a menu set then finding compensations, but reduces overfitting. We do this by including a population of fixed training scenarios in addition to variable training scenarios. These *fixed LCF scenarios* provide a sample of unbiased driver selections that cannot be strategically manipulated by the solver to only accept requests when they are assigned so that the required incentives for driver participation are minimized (as the solver does with variable LCF scenarios). The fixed LCF scenarios are composed of parameter values, denoted as $\hat{y}^{\hat{s}}_{ij}$, to represent driver behaviour, similar to the y^s_{ij} variables. For each fixed LCF scenario $\hat{s}$ in a sample set of fixed scenarios $\hat{\Omega}$, $\hat{y}^{\hat{s}}_{ij} = 1$ if $x^*_{ij} = 1$ from the returned SLSF menu and if driver j is willing to accept request i in scenario $\hat{s}$, and $\hat{y}^{\hat{s}}_{ij} = 0$ otherwise. The fixed LCF scenarios are generated under the same fixed compensations used to generate SLSF training scenarios to obtain $\boldsymbol{x}^*$. Because we equally weight the variable LCF scenarios, we also equally weight the fixed LCF scenarios instead of basing the weights on the fixed scenarios' likelihoods of occurring.

To incorporate the fixed LCF scenarios into our formulation, we introduce new variables $\hat{\boldsymbol{v}}$ and $\hat{\boldsymbol{\phi}}$ to capture the recourse assignment ($\hat{\boldsymbol{v}}$) and assignment multiplied by the $\boldsymbol{u}$ component of the compensation ($\hat{\boldsymbol{\phi}}$) of the fixed LCF scenarios. Next, we augment all summation terms that sum across the variable scenarios in the objective function (2) and constraint (9) to instead sum across both variable and fixed LCF scenarios. Lastly, for the constraints that use v^s_{ij} and/or ϕ^s_{ij} over a single scenario, that

is, (5)-(7), (12)-(13), and the $\boldsymbol{v}$ component of (15), we add a copy of these constraints that applies to the fixed LCF scenario counterpart parameter/variable. The constraints that only use y^s_{ij}, that is, (4) and the y^s_{ij} component of (15), don't need a fixed scenario version as these relationships are always met because of how the $\hat{y}^{\hat{s}}_{ij}$ are generated. We refer to this two-step solution method with Fixed Menus and Scenarios as SOL-FMS, and this formulation's second step (optimizing compensation), LCF-FMS, is given by (16)-(23), (4)-(8),(10)- (13), and (15).

$$\max_{\substack{\ddot{p},u,v,\\ \hat{v},y,\phi,\\ \hat{\phi}}} \quad \frac{1}{|\Omega|+|\hat{\Omega}|}\sum_{i\in M}\sum_{j\in N}\left(\sum_{s\in\Omega}\left((\hat{c}_{ij}-\alpha_{ij})\,v^s_{ij}-\phi^s_{ij}\right)+\sum_{\hat{s}\in\hat{\Omega}}\left((\hat{c}_{ij}-\alpha_{ij})\,\hat{v}^{\hat{s}}_{ij}-\hat{\phi}^{\hat{s}}_{ij}\right)\right) \tag{16}$$

s.t. [Constraints (4)-(8),(10)- (13), (15)]

$$\hat{v}^{\hat{s}}_{ij}\le\hat{y}^{\hat{s}}_{ij} \qquad \forall\hat{s}\in\hat{\Omega},\quad \forall i\in M,\quad \forall j\in N \tag{17}$$

$$\sum_{i\in M}\hat{v}^{\hat{s}}_{ij}\le 1 \qquad \forall\hat{s}\in\hat{\Omega},\quad \forall j\in N \tag{18}$$

$$\sum_{j\in N}\hat{v}^{\hat{s}}_{ij}\le 1 \qquad \forall\hat{s}\in\hat{\Omega},\quad \forall i\in M \tag{19}$$

$$\frac{1}{|\Omega|+|\hat{\Omega}|}\left(\sum_{s\in\Omega}y^s_{ij}+\sum_{\hat{s}\in\hat{\Omega}}\hat{y}^{\hat{s}}_{ij}\right)\le prem\cdot\ddot{p}_{ij} \qquad \forall i\in M,\quad \forall j\in N \tag{20}$$

$$0\le\hat{\phi}^{\hat{s}}_{ij}\le u_{ij} \qquad \forall\hat{s}\in\hat{\Omega},\quad \forall i\in M,\quad \forall j\in N \tag{21}$$

$$\beta_{ij}\hat{v}^{\hat{s}}_{ij}+u_{ij}-\beta_{ij}\le\hat{\phi}^{\hat{s}}_{ij}\le\beta_{ij}\hat{v}^{\hat{s}}_{ij} \qquad \forall\hat{s}\in\hat{\Omega},\quad \forall i\in M,\quad \forall j\in N \tag{22}$$

$$\hat{v}^{\hat{s}}_{ij}\in\{0,1\} \qquad \forall\hat{s}\in\hat{\Omega},\quad \forall i\in M,\quad \forall j\in N \tag{23}$$

This new formulation creates a potential source of infeasibility. The willingness constraint (20) has some minimum amount of driver participation for each driver-request pair in a menu, dictated by the fixed $\hat{y}^{\hat{s}}_{ij}$. The $\hat{y}^{\hat{s}}_{ij}$ are generated via a random process, and it's possible that the compensation $\alpha_{ij}+u_{ij}$ required to achieve that participation per constraints (8) and (20) may be higher than the upper bound $\alpha_{ij}+\beta_{ij}$. That is, for any driver j-request i pair in the menu set, even if a solution includes no additional driver participation in the variable scenarios, i.e. $\sum_{s\in\Omega}y^s_{ij}=0$, we have via (8) and (20) that

$$\frac{1}{|\Omega|+|\hat{\Omega}|}\sum_{\hat{s}\in\hat{\Omega}}\hat{y}^{\hat{s}}_{ij}\le\frac{prem\cdot wage_{min}}{\alpha_{ij}(wage_{max}-wage_{min})}u_{ij},$$

which means that

$$\frac{\alpha_{ij}(wage_{max}-wage_{min})}{(|\Omega|+|\hat{\Omega}|)\cdot prem\cdot wage_{min}}\sum_{\hat{s}\in\hat{\Omega}}\hat{y}^{\hat{s}}_{ij}\le u_{ij}.$$

If $\left(\alpha_{ij}(wage_{max}-wage_{min})/((|\Omega|+|\hat{\Omega}|)\cdot prem\cdot wage_{min})\right)\cdot\sum_{\hat{s}\in\hat{\Omega}}\hat{y}^{\hat{s}}_{ij}$ is higher than the u_{ij} upper bound β_{ij}, this renders the formulation infeasible. We can therefore prevent any infeasibility by requiring that β_{ij} for any driver request pair in a menu is at least $\left(\alpha_{ij}(wage_{max}-wage_{min})/((|\Omega|+|\hat{\Omega}|)\cdot prem\cdot wage_{min})\right)\cdot\sum_{\hat{s}\in\hat{\Omega}}\hat{y}^{\hat{s}}_{ij}$.

3.2.4 SOL-IT, Iterative Solution Method

We make an algorithmic augmentation to SOL-FMS by executing it iteratively, alternating between compensations being parameters input to optimize the menu set, and then fixing the menu set and optimizing compensations. Algorithm A.1 in Appendix A summarizes each step of this process. We begin with finding the optimal menu set using SLSF under a default fixed compensation scheme, then the compensations are optimized using LCF-FMS. This is considered to be one iteration. A new personalized (fixed) compensation is determined for the next iteration by starting with the fixed compensations used for SLSF in the latest iteration. For any driver-request pairs that are in the current menu and have a compensation above the lower bound in the latest optimized compensation solution, i.e. $u_{ij} > 0$, the new compensation $\alpha_{ij} + u_{ij}$ replaces the old compensation. This updated compensation set is then used to generate training scenarios for SLSF to determine a new menu set, and again to generate fixed LCF scenarios to be inputted along with the new menu set into LCF-FMS to determine the new optimal compensations. We refer to this iterative solution method as SOL-IT.

We show in Section 4.2 that iterating does allow us to improve the quality of our solutions in terms of objective value and the number of matched requests, yet, this improvement is not monotonic with each iteration and the improvement between iterations varies between different parameter instances (trip data). Thus, returning the solution produced by the final iteration may not perform as well as another menu/compensation solution produced in an earlier iteration. Further complicating which solution to select is the fact that objective values produced by SAA are inherently positively biased, as we are sampling scenarios; thus, we cannot simply select the solution with the highest returned objective value either. To determine the quality of a solution, we use a separate set of random *test scenarios* to analyze the performance of the decision variable values found by the program. These random test scenarios provide an unbiased performance estimate, and details of this process are presented in Section 3.3. We show in Horner *et al.* (2021) that quality performance analysis of a solution can take significant time compared to the time needed to solve the formulations in each iteration, so running quality performance analysis for every iteration's solution would be time consuming. As a compromise, for each iteration we conduct a smaller, less accurate but less time-consuming performance analysis. These tests use the same performance analysis method but only examine a small random sample of test scenarios instead of a much larger sample used for higher-quality performance analysis. Specifically, the best-found solution returned by the formulation is initialized as the initial compensation scheme and the menu set returned by SLSF in the first iteration. This solution is the same solution that would be returned by only using SLSF from Horner *et al.* (2021) instead of SOL-IT. We want our returned solution to perform at least as well as this solution, so we use it as a baseline comparison in our small performance analysis tests and the SLSF solution is returned if no solution in the iterative method has a higher objective estimate than this solution. After LCF-FMS is run for SOL-IT in the first iteration and after each subsequent iteration, the new menu set and compensation values undergo the small performance analysis test. If the objective estimate is higher than that of the current best-found solution, the current iteration's solution replaces the best-found solution.

3.3 Performance analysis

All of our solution methods (see Figure 2) return menu sets, compensations, and the corresponding objective value. However, this objective value can be overfitted to the variable and/or fixed LCF training scenarios inputted, e.g., even the objective values of a standard SAA problem with i.i.d. scenarios has bias (Kleywegt *et al.*, 2002). For this reason we create an estimate of a solution's performance without any inherent bias, and we do this by conducting a separate round of analysis that calculates the expected performance across a set of randomly generated *test scenarios*. Performance analysis is conducted on a given solution, and because the menu set is provided in the solution, each test scenario only needs to have driver selections for each driver-request pair in a menu set. We use $\dot{s}$ to refer to a test scenario and store the scenario information in $\dot{\boldsymbol{y}}$ values. Given solution menu set $\boldsymbol{x}^*$, we let $\dot{y}_{ij}^{\dot{s}} = 1$ if $x_{ij}^* = 1$ and driver j is willing to accept request i in scenario $\dot{s}$ and equals zero otherwise. The sample set of these performance analysis test scenarios is denoted as $\dot{\Omega}$.

For each test scenario of driver selections $\dot{s} \in \dot{\Omega}$, we find the optimal recourse assignment $\dot{\boldsymbol{v}}$ and corresponding objective value for a solution $[\boldsymbol{comp}^*, \boldsymbol{x}^*]$. For our solution methods, the returned solution includes $\boldsymbol{x}^*$, and $\boldsymbol{comp}^*$ can be calculated using $\boldsymbol{u}^*$ from the solution, as $\boldsymbol{comp}^* = \boldsymbol{\alpha} + \boldsymbol{u}^*$. Our computational experiments also run performance analysis on SLSF and deterministic methods for comparison, in which case $\boldsymbol{x}^*$ is the returned solution and $\boldsymbol{comp}^*$ is part of the fixed parameter inputs.

The objective value of the optimal recourse assignment for each scenario is calculated, and a weighted average of the objective values is returned as the performance estimate for the solution, where the weight for each scenario is the likelihood of the scenario divided by the sum of likelihoods of all scenarios in $\dot{\Omega}$. The likelihood of a scenario is easy to calculate, as each $\dot{y}_{ij}^{\dot{s}}$ value is generated in an independent Bernoulli distribution, with $\mathbb{P}(\dot{y}_{ij}^{\dot{s}} = 1) = x_{ij}^* p_{ij}^*$. Here p_{ij}^* follows the willingness formula (1) based on the solution compensation, meaning $p_{ij}^* = \min\{\ddot{p}_{ij}^*, 1\}$. For an SLSF/deterministic solution, p_{ij}^* can be calculated using $\boldsymbol{comp}^*$, $\boldsymbol{\alpha}$, and (1). The likelihood of test scenario $\dot{s}$ is given by

$$\mathbb{P}(\dot{s}) = \left(\prod_{i,j:x_{ij}^*=1, \dot{y}_{ij}^{\dot{s}}=1} p_{ij}^* \right) \left(\prod_{i,j:x_{ij}^*=1, \dot{y}_{ij}^{\dot{s}}=0} (1 - p_{ij}^*) \right). \tag{24}$$

In addition to considering profit (and any additional reward components such as matching bonuses), we also want to prioritize driver experience, namely, reducing the chance a driver accepts one or more requests but does not receive an assignment, making them an *unhappy driver*. This negative interaction, especially if experienced repeatedly, can result in drivers leaving the platform. We introduce a penalty parameter $\boldsymbol{d}$, where d_{ij} can depend on $fare_i$ that unhappy driver j would have received if they were assigned request i, and/or driver j's history with the platform including how recently they have been an unhappy driver, etc. The binary variable set $\dot{\boldsymbol{z}}$ is used to keep track of the unhappy driver experience, where $\dot{z}_{ij}^{\dot{s}} = 1$ if request i is accepted by driver j in scenario $\dot{s}$ but driver j receives no assignment in scenario $\dot{s}$, and equals zero otherwise. With this in mind, the optimal recourse assignment and corresponding objective value for test scenario $\dot{s}$ is found by optimizing the linear program (25)-(31).

The objective function (25) simultaneously maximizes the total non-cost utility ($\hat{c}_{ij}$) minus the cost

paid to driver ($comp^*_{ij}$) of the assignment, while minimizing the unhappy driver penalties (d_{ij}) incurred by the assignment. Constraints (26)-(28) and (31) are adapted from our other formulations to create a valid assignment, ensuring requests are assigned only if they are accepted, each driver receives at most one request, a request is assigned to at most one driver, and that our $\dot{v}^{\dot{s}}_{ij}$ are binary, respectively. Constraints (29)-(30) enforce that the $\dot{z}^{\dot{s}}_{ij}$ take on the correct value based on the assignment and the drivers' responses: constraint (29) uses the logical statement that $\dot{z}^{\dot{s}}_{ij} \geq 1$ if and only if request i is in driver j's menu (i.e. $x^*_{ij} = 1$), driver j accepts the request in test scenario $\dot{s}$ (i.e. $\dot{y}^{\dot{s}}_{ij} = 1$), and driver j receives no assignment in test scenario $\dot{s}$ (i.e. $1 - \sum_{k \in M} \dot{v}^{\dot{s}}_{kj} = 1$). For all other possible outcomes, $\dot{z}^{\dot{s}}_{ij} \geq 0$ or $\dot{z}^{\dot{s}}_{ij} \geq -1$ by (29) depending on the value of x^*_{ij}, $\dot{y}^{\dot{s}}_{ij}$ and $\sum_{k \in M} \dot{v}^{\dot{s}}_{kj}$. The optimization problem minimizes these penalties and therefore sets each $\dot{z}^{\dot{s}}_{ij}$ to its minimum value, which because of constraint (30), is always zero or one as desired. This also means the binary nature of $\dot{z}^{\dot{s}}_{ij}$ is naturally enforced so a binary constraint is not necessary.

$$\max_{\dot{v},\, \dot{z}} \quad \sum_{i \in M} \sum_{j \in N} \left((\hat{c}_{ij} - comp^*_{ij}) \dot{v}^{\dot{s}}_{ij} - d_{ij} \dot{z}^{\dot{s}}_{ij} \right) \tag{25}$$

$$\text{s.t.} \quad \dot{v}^{\dot{s}}_{ij} \leq \dot{y}^{\dot{s}}_{ij} \qquad \forall i \in M, \quad \forall j \in N \tag{26}$$

$$\sum_{i \in M} \dot{v}^{\dot{s}}_{ij} \leq 1 \qquad \forall j \in N \tag{27}$$

$$\sum_{j \in N} \dot{v}^{\dot{s}}_{ij} \leq 1 \qquad \forall i \in M \tag{28}$$

$$\dot{z}^{\dot{s}}_{ij} \geq -2 + \dot{y}^{\dot{s}}_{ij} + x^*_{ij} + (1 - \sum_{k \in M} \dot{v}^{\dot{s}}_{kj}) \qquad \forall i \in M, \quad \forall j \in N \tag{29}$$

$$\dot{z}^{\dot{s}}_{ij} \geq 0 \qquad \forall i \in M, \quad \forall j \in N \tag{30}$$

$$\dot{v}^{\dot{s}}_{ij} \in \{0, 1\} \qquad \forall i \in M, \quad \forall j \in N \tag{31}$$

Our SLSF formulation also has unhappy driver variables, $\bar{\boldsymbol{z}}$, with equivalent constraints to (29)-(30) and an equivalent objective function penalty (see Appendix B). This means that the solutions from SLSF and therefore solutions from SOL-FM, SOL-FMS, and SOL-IT have menus that take the effect of unhappy drivers into account and are designed to minimize this penalty across the SLSF training scenarios. With Lemma 3.1 and Theorem 3.1, we claim that reformulating LCF to incorporate an unhappy driver variable set $\boldsymbol{z}$ with equivalent versions of constraints (29)-(30) and the unhappy driver penalty terms in the objective (25), resulting in the formulation *LCF-withZ*, contributes nothing to the solution. If all unhappy driver penalties (the d_{ij}) are positive then the optimal solution will never have any unhappy drivers (Lemma 3.1); moreover, the optimal objective value and first-stage decisions (menus and compensations) are not affected by the incorporation of unhappy drivers (Theorem 3.1). This is because driver responses are variable so the solver can have any driver that would be unhappy in a scenario simply not accept any requests in the scenario.

Lemma 3.1. *Any optimal solution of LCF-withZ can only have positive $z^{*\dot{s}}_{ij}$ values for $i \in M$ and $j \in N$ such that $d_{ij} = 0$.*

Theorem 3.1. *LCF and LCF-withZ share the same optimal objective value and set(s) of optimal decision variables ($\boldsymbol{x}$ and $\boldsymbol{u}$).*

The proofs of Lemma 3.1 and Theorem 3.1, along with a formal definition of LCF-withZ are deferred to Appendix C. While LCF-FMS does have the capacity to non-trivially model unhappy drivers, as it has some number of fixed driver selection scenarios, we do not add unhappy driver variables to this model as it greatly increases the complexity of the formulation. In addition, directly capturing unhappy driver variables for these cases is only partially possible, that is, it is possible to capture them only for the fixed scenarios, but not for the variable scenarios.

3.4 Scenario Type Summary

Between our LCF solution methods and performance analysis, we use four different types of scenarios: variable LCF (training) scenarios in all LCF solution methods, fixed SLSF training scenarios generated as inputs to SLSF, fixed LCF (training) scenarios generated as inputs to LCF-FMS, and fixed test scenarios generated for performance analysis. The notation and additional details about the different scenario types are summarized in Table 2.

Table 2: Information about the different scenario types. Column 1 displays the scenario type, Column 2 the notation of an individual scenario of each type, Column 3 the notation for the sample set of scenarios, Column 4 the binary parameter (or variable for variable LCF scenarios) set used that contains the driver selection information. In Column 5, an × signifies that this scenario type has driver behavior information for all driver-request pairs, while a ✓ signifies that driver selections are only considered for pairs in the menu set. Column 6 displays the scenario generation method for the fixed scenario types. The last column displays the weighting formula used between scenarios (which is then normalized to sum to one).

Type	Scen	Samp Set	Par/Var	$x_{ij}=1$?	Generation	Weight
variable LCF (training)	s	Ω	$\boldsymbol{y}^s$	✓	×	equal
SLSF (training)	$\bar{s}$	$\bar{\Omega}$	$\bar{\boldsymbol{y}}^{\bar{s}}$	×	mutated	Eqn. (32)
fixed LCF (training)	$\hat{s}$	$\hat{\Omega}$	$\hat{\boldsymbol{y}}^{\hat{s}}$	✓	random	equal
Test (performance)	$\dot{s}$	$\dot{\Omega}$	$\dot{\boldsymbol{y}}^{\dot{s}}$	✓	random	Eqn. (24)

For our fixed scenario types, we generate the driver behavior, $\bar{\boldsymbol{y}}^{\bar{s}}/\hat{\boldsymbol{y}}^{\hat{s}}/\dot{\boldsymbol{y}}^{\dot{s}}$ values, using one of two methods. For performance analysis, we generate test scenarios randomly to avoid bias in the estimate. This is easily done by generating $\dot{y}^{\dot{s}}_{ij}$ via a Bernoulli random process with success probability p^*_{ij} for each (i,j) pair such that $x^*_{ij}=1$, where $\boldsymbol{x}^*$ is the solution menu and $\boldsymbol{p}^*$ is calculated using the solution's compensations (fixed or optimized). We also use random generation for the fixed LCF training scenarios, generating $\hat{y}^{\hat{s}}_{ij}$ via a Bernoulli random process with success probability p_{ij} for each (i,j) pair such that $x^*_{ij}=1$, where $\boldsymbol{x}^*$ is the menu produced by SLSF (for SOL-IT this is the most recently-produced SLSF menu) and $\boldsymbol{p}$ is the driver selection probabilities corresponding to the fixed compensations used to find that SLSF menu. However, for the SLSF training scenarios, we need to use another method instead of random generation. This is because SLSF training scenarios are weighted based on their relative likelihood. An SLSF

training scenario $\bar{s}$ has likelihood given by formula (32), which is a large exponential function, so the range of scenario likelihood is scattered between several different orders of magnitude. Among a random sample of SLSF training scenarios, after normalizing the likelihood weights so that they sum to one, the weights are dominated by the few most likely scenarios, enough so that the solver may assign trivial values to variables corresponding to the other scenarios. For this reason we use a sampling method that creates a sample set of SLSF training scenarios that have similar likelihoods. These scenarios are created by mutating the most likely scenario into different scenarios, which we refer to as 'mutated' scenarios. A more detailed description of mutated scenario generation is provided in Appendix D.

$$\mathbb{P}(\bar{s}) = \left(\prod_{i,j:\bar{y}^{\bar{s}}_{ij}=1} p_{ij} \right) \left(\prod_{i,j:\bar{y}^{\bar{s}}_{ij}=0} (1 - p_{ij}) \right) \tag{32}$$

As shown in column 5 in Table 2, the information needed to define a scenario depends on the scenario type. For SLSF training scenarios, the menu is unknown so a $\bar{y}^{\bar{s}}_{ij}$ driver selection value is generated for every $i \in M$ and $j \in N$. In contrast, for fixed LCF scenarios and performance analysis test scenarios, a menu has been previously determined and therefore a $\hat{y}^{\hat{s}}_{ij}/\dot{y}^{\dot{s}}_{ij}$ value is only needed for driver j-request i pairs contained in that menu set and $\hat{y}^{\hat{s}}_{ij}/\dot{y}^{\dot{s}}_{ij}$ is set to be zero for the remaining pairs as their values do not affect the solution. For variable LCF scenarios, though the y^{s}_{ij} values are variable, constraint (4) enforces that y^{s}_{ij} values for pairs not in the menu are zero (whether the menu is variable or fixed).

4 Computational Results

4.1 Data Input and Generation

Our computational results are generated using the Chicago Regional transportation network, which contains 12,982 nodes and 39,018 single-directional arcs that approximate the roads and traffic of the Chicago Region (Chicago Area Transportation Study (CATS), 2016). Each arc contains information including the length of the arc and the time needed to traverse it when the traffic on all arcs follow the equilibrium traffic flow. The data set contains origin-destination demand quantities between pairs of nodes that represent zone centroids, and we focus on a subgraph with 350 centroids that approximates Chicago Proper. Using these demand quantities as probability weights, we generate driver and request trips, with the origin-destination (OD) pair of a trip being a randomly selected centroid pair. We focus on occasional drivers, in which each driver has their own trip they plan to make for their own purposes, and when matched they reroute their trip to accommodate the request. Because the OD demand in the network represents the travel demand of people rather than parcels, we consider our experiments to be implemented in a ridesharing context. Using the arc information along with Dijkstra's Algorithm, we calculate the duration and length of the shortest trip (we minimize path travel time rather than path length) between any two centroids.

Table 3 displays the formulas for our parameters. The total fee collected for each matched request is calculated using Chicago's Lyft rates: a fixed \$1.85 booking fee is collected and paid directly to the plat-

form, and then a variable fare, traditionally shared with the driver, is calculated based on the time and distance of the trip, with a minimum fare of $3 (EstimateFares.com, 2019). The platform's parameters for all utilities excluding cost paid to drivers, $\hat{\boldsymbol{c}}$, contains the fare plus the fixed booking fee, a matching bonus, and a per-minute rider waiting penalty (i.e. the time required for the driver to arrive at the rider's location). To represent the importance of requests being fulfilled, we consider a default match bonus $match_i$ of 5 for any request. Depending on a platform's service commitment, the platform could vary this value as well as tailor $match_i$ to each request i to, for example, to assign different match priorities to requests depending on the amount of time the rider has been waiting for a match. The per-minute wait penalty is derived from the per-minute fee in the fare calculation, multiplied by 0.2 to represent the platform's portion of the minute fee if they were to pay 80% of the fare to the driver. The maximum driver compensation we allow is the full fare plus the booking fee plus an additional $5 as it may be possible for the platform to lose money on one request but gain a higher total objective value/profit (due to the match bonus which is a surrogate for demand-side experience with the platform). If this compensation upper bound is lower than α_{ij}, then β_{ij} is set to zero for feasibility. Lyft advertises that drivers receive 80% of the fare not including the booking fee, so we require that the minimum compensation to drivers $mincomp_i$ is at least 80% of the fare and is also at least $4 out of consideration for the driver (Helling, 2019). These minimum compensation values are also used as the fixed compensation scheme for SLSF, both when SLSF is used as a standalone method and when it is used in SOL-FM, SOL-FMS, and SOL-IT (for SOL-IT they are the initial SLSF compensations which are then updated each iteration). The unhappy driver penalty d_{ij} used for SLSF and performance analysis is assumed to depend on driver j's participation history with the platform, for example how recently they have been an unhappy driver. To simulate this information we generate d_{ij} under a random uniform distribution. How to set suitable values for the premium parameter $prem$, the number of variable and fixed LCF scenarios, and the number of iterations, are explored as experiments in Section 4.2.

Table 3: Formulas for the input parameters. LCF-FMS (in both SOL-FMS and SOL-IT) uses a different equation to calculate β_{ij} than LCF and LCF-FM, and we first display the simpler β_{ij} formula for LCF and LCF-FM followed by the β_{ij} formula for LCF-FMS.

Parameter	**Formula**
$fare_i$	$\max\left\{\left(1.79+0.28\cdot mins_i+0.81\cdot miles_i\right),\ 3\right\}$
$\hat{c}_{ij}$	$1.85+fare_i+match_i-0.056\cdot mins_{ij}$
α_{ij}	$wage_{min}\cdot effort_{ij}$
β_{ij}(for LCF, LCF-FM)	$\max\{fare_i+1.85+5-\alpha_{ij},0\}$
β_{ij} (for LCF-FMS)	$\max\{fare_i+1.85+5-\alpha_{ij},\frac{\alpha_{ij}(wage_{max}-wage_{min})}{\lvert\Omega+\hat{\Omega}\rvert\cdot prem\cdot wage_{min}}\sum_{\hat{s}\in\hat{\Omega}}\hat{y}_{ij}^{\hat{s}}\}$
$mincomp_i$	$\max\{0.8\cdot fare_i,4\}$
d_{ij}	$fare_i+r_j,\qquad$ where $r_j\sim\mathcal{U}([0,3])$

The menu size is allowed to range between zero and five, that is, $\ell=0$ and $\theta=5$. We let $effort_{ij}$ be driver j's extra driving time in hours if they fulfill request i, so p_{ij} is a function of the driver's wage

per hour of extra driving. We set the p_{ij} piecewise breakpoints to be $wage_{min} = \$10/hr$ and $wage_{max} = \$25/hr$. Using these parameters and the Chicago network, we generate 100 problem instances, each with 20 requests and 20 drivers. Each problem instance contains the origins and destinations of the drivers and requests and the randomly-generated information for the d_{ij} and for 100 SLSF training scenarios. In Horner *et al.* (2021), the effect of SLSF training scenario sample size on the quality of the solution was tested to determine that 100 samples provides a well-performing solution in a reasonable amount of time. Accordingly, our SLSF solutions produced (standalone or in a solution method) use 100 SLSF training scenarios. The SLSF solution corresponding to each problem instance's generated SLSF training scenarios is determined, and using the resulting menu set, a set of 50 fixed LCF training scenarios is generated for each problem instance to be used in any experiment using SOL-FMS and the first iteration of SOL-IT. For fair comparisons, all computational experiments are conducted over these same 100 problem instances. In Horner *et al.* (2021), using a set of 5,000 random test scenarios provided a sufficiently accurate performance estimate of a menu set with given compensations for our chosen number of riders and drivers. With this in mind, all performance analysis tests are conducted using 5,000 random test scenarios. The small performance analysis tests conducted in SOL-IT each use 200 random test scenarios. All experiments are completed in MATLAB R2019a using CPLEX 12.9 on an i7-core 1.9GHz CPU with 32GB RAM. For all experiments, the SLSF solver terminates upon finding a solution with an objective value within 1% of the optimal or by reaching the 60 second time limit imposed on the solver. All LCF, LCF-FM, and LCF-FMS solvers terminate upon finding a solution with an objective value within 5% of the optimal or after 300 seconds have elapsed.

4.2 Solution Method Setting Recommendations

In this section we conduct experiments to determine recommended setting configurations for implementing our SOL-IT method, namely, the value of $prem$, as well as the number of variable and fixed LCF scenarios, and the number of iterations. We refer to settings using SOL-IT with A iterations, B variable scenarios, and C fixed scenarios as SOL-IT$_A(B,C)$. For each specified setting in an experiment, the solution is found for the same 100 problem instances and performance analysis is performed for each solution. We then consider the average performance analysis objective value of each setting, and the averages of three additional performance metrics of interest that individually affect the performance analysis objective estimate: profit, the number of unmatched requests, and the number of requests accepted by an unhappy driver, that is, the expected number of nonzero $\dot{z}^{\hat{s}}_{ij}$ values in a given test scenario. The average number of unmatched requests is not explicitly part of the objective function (16), but is rather the absence of receiving the match bonus $match_i$ for the unfulfilled requests. In addition to these four solution performance metrics, we include the average runtime for each setting.

First, we compare five premium values, spaced equally in the range $[0.5, 1]$ using SOL-IT$_{10}(5,5)$, with the results displayed in Table 4. The premium setting $prem = 0.75$ has the best objective value, unmatched request rate, and runtime average out of the five settings. It has a low unhappy driver request average, second only to $prem = 0.625$, which has worse averages than $prem = 0.75$ in every other metric. Additionally, 0.5 and 0.625 are not ideal premium settings as the first iteration's SLSF solution is returned

as the best-found solution in 87 and 39 out of the 100 problem instances, respectively. The long runtime needed for $prem = 0.5$ is due to the LCF-FMS solver often timing out at the 300 second time limit before reaching the cutoff solution optimality gap. Given this performance, we select $prem = 0.75$ as our recommended setting and use this premium value for the remainder of our experiments.

Table 4: Performance averages of SOL-IT$_{10}(5,5)$ with different premium values. Column 1 displays the premium setting, while column 2 displays the average performance analysis objective value (Obj) across the 100 problem instances. Columns 3, 4, 5, and 6 display the average number of unmatched requests (Unmat), expected number of requests accepted by an unhappy driver (Unhap), the expected profit (Prof), and the average runtime (Time) in seconds, respectively.

	Performance				
	Obj	Unmat	Unhap	Prof ($)	Time (s)
$prem$ =0.5	161.34	1.320	0.134	81.74	643.7
$prem$ =0.625	162.54	0.825	0.094	80.05	236.3
$prem$ =0.75	164.14	0.687	0.130	81.45	200.7
$prem$ =0.875	163.58	0.963	0.150	82.42	226.0
$prem$ =1	163.04	1.064	0.165	82.76	239.2

We next examine different variable and fixed LCF scenario sample sizes. Each tested setting uses 10 SOL-IT iterations. The averages over the 100 problem instances for the four solution performance metrics and runtime are given in Table 5. Many of these configuration settings produce similar average performance values, suggesting robustness of our solution method to scenario sample sizes. In addition, no pair of variable and fixed scenario sample sizes in Table 5 is the best or the worse for all of the solution quality metrics. Yet, settings with an unequal number of variable and fixed scenarios (SOL-IT$_{10}(10,20)$, SOL-IT$_{10}(10,50)$, SOL-IT$_{10}(20,10)$, and SOL-IT$_{10}(50,10)$) tend to have a lower objective average, and settings with an equal number of variable and fixed scenarios (SOL-IT$_{10}(5,5)$, SOL-IT$_{10}(10,10)$, SOL-IT$_{10}(20,20)$, and SOL-IT$_{10}(50,50)$) tend to perform better on the solution performance metrics.

Table 5: Performance averages of SOL-IT with different variable and fixed scenario sample sizes. Column 1 displays the setting configuration, while column 2 displays the average objective value. Columns 3, 4, 5, and 6 display the average number of unmatched requests, the expected number of requests accepted by an unhappy driver, the expected profit, and the average runtime, respectively.

	Performance				
	Obj	Unmat	Unhap	Prof ($)	Time (s)
SOL-IT$_{10}(5,5)$	164.14	0.687	0.130	81.45	200.7
SOL-IT$_{10}(10,10)$	164.08	0.807	0.141	81.99	278.9
SOL-IT$_{10}(10,20)$	163.44	0.693	0.093	80.25	328.7
SOL-IT$_{10}(10,50)$	161.11	1.288	0.135	81.30	301.5
SOL-IT$_{10}(20,10)$	163.30	0.966	0.164	82.28	368.5
SOL-IT$_{10}(20,20)$	164.08	0.784	0.142	81.91	440.4
SOL-IT$_{10}(50,10)$	163.62	1.008	0.148	82.55	600.4
SOL-IT$_{10}(50,50)$	164.36	0.736	0.135	81.80	1580.1

The performance results vary between different problem instances, so to gain a more thorough understanding of the relative performance between the different setting configurations, we conduct paired sample t-tests. For all paired sample t-tests conducted in our experiments, if some solution performance metric for setting A (listed first) is being compared to setting B (listed second), the null hypothesis is that the population mean for that metric is equal for both settings, and the alternate hypothesis is that setting A has a 'better' population average than setting B. That is, for the objective average or profit average, setting A would have a higher population average, while the number of unmatched requests or unhappy driver requests would have a lower population average for setting A. If the p-value is above a $p = 0.05$ significance cutoff, we fail to reject the null hypothesis that both settings produce an equal mean value.

We examine p-values for each of the four solution performance metrics in Table 6, comparing the best metric average out of the settings to each other setting. The second-best average is also compared to each setting for unhappy driver requests and profit because the best setting is statistically better than all other settings for these metrics. SOL-IT$_{10}$(5,5) and SOL-IT$_{10}$(50,50) are statistically within range of having the best objective value and unmatched requests (that is, the t-test does not yield a statistical difference), and statistically may have the second best number of unhappy driver requests. This is the best track record among the tested scenario sample sizes in Table 5, with the next best sizes being SOL-IT$_{10}$(10,10) and SOL-IT$_{10}$(20,20). Yet, SOL-IT$_{10}$(50,50) takes almost eight times as long as SOL-IT$_{10}$(5,5). Though SOL-IT$_{10}$(50,50) does have a higher profit, no other statistically significant difference exist between their performance in other metrics. The magnitude of the difference in profit does not make up for the magnitude in difference for the runtime, making SOL-IT$_{10}$(5,5) our recommended setting.

Table 6: Paired sample t-test p-values for different solution performance metrics and settings. SOL-IT is shortened to 'S.' for the row and column labels. Each column displays the resulting p-values from comparing the specified metric of one setting A to each of the other settings from Table 5 as setting B. All p-values above the 0.05 threshold are in bold. The relative rank of setting A is also displayed, where rank 1 means that setting A has the best performance average for the metric out of the settings in Table 5. Rank 2 means it has the second best average. A p-value below 0.00005 is displayed as 0.

	Metric	Obj	Unmat	Unhap	Unhap	Prof ($)	Prof ($)
	Set. A	S.$_{10}$(50,50)	S.$_{10}$(5,5)	S.$_{10}$(10,20)	S.$_{10}$(5,5)	S.$_{10}$(50,10)	S.$_{10}$(20,10)
Setting B	Rank	1	1	1	2	1	2
S.$_{10}$(5,5)		**0.1275**	–	0.0008	**–**	0	0
S.$_{10}$(10,10)		**0.0953**	0.0030	0.0002	**0.1744**	0	0.0147
S.$_{10}$(10,20)		0	**0.4526**	–	**0.9992**	0	0
S.$_{10}$(10,50)		0	0	0.0006	**0.3357**	0.0003	0.0040
S.$_{10}$(20,10)		0.0001	0	0	0.0081	0.0282	–
S.$_{10}$(20,20)		**0.0658**	0.0083	0	**0.1367**	0	0.0047
S.$_{10}$(50,10)		0.0008	0	0	0.0335	–	**0.9718**
S.$_{10}$(50,50)		–	**0.1231**	0.0005	**0.3251**	0	0.0005

All testing thus far uses 10 iterations, so we next examine the effect that the number of iterations has on performance. Table 7 displays the metric averages and p-values of several settings when compared to SOL-IT$_{10}$(5,5) (i.e., setting A is SOL-IT$_{10}$(5,5)). We test for iterations both higher and lower than 10

to see if performance can be improved with more iterations or if similar performance can be achieved with fewer iterations that requires less runtime. These iteration experiments demonstrate the gains from iterating and when the benefits begin to plateau. For our iteration reduction experiments, shown above the lower horizontal line in Table 7, we test the most promising variable and fixed scenario sample size settings out of the settings tested in Tables 5 and 6 (SOL-IT$_*$(5,5), SOL-IT$_*$(10,10), SOL-IT$_*$(20,20), and SOL-IT$_*$(50,50)) with 1, 2, 5, and 7 iterations. When only one or two iterations are implemented, all settings have significantly lower objective values and more unmatched requests than SOL-IT$_{10}$(5,5). In addition, many settings have more unhappy driver requests than SOL-IT$_{10}$(5,5), though the profit average is higher. As the number of iterations increases, the objective, number of unmatched requests, and number of unhappy driver requests improve, while profit decreases. At seven iterations, SOL-IT$_7$(10,10) and SOL-IT$_7$(50,50) are statistically within range of the SOL-IT$_{10}$(5,5) objective value and number of unhappy driver requests, and both have a statistically higher profit (the p-value comparing SOL-IT$_7$(10,10) to SOL-IT$_{10}$(5,5) is <0.00005, and comparing SOL-IT$_7$(50,50) to SOL-IT$_{10}$(5,5) the p-value is 0.0006). However, both have a statistically higher number of unmatched requests and neither offers a significant decrease in runtime (with SOL-IT$_7$(50,50) having a much higher runtime). For iteration extension, we test SOL-IT$_{13}$(5,5), SOL-IT$_{13}$(10,10), and SOL-IT$_{13}$(20,20) to show the effect of an additional three iterations. SOL-IT$_{13}$(50,50) is not tested as even with 10 iterations the runtime requirements are significantly higher than the other sizes. The results below the lower horizontal line in Table 7 show that all three of the 13-iteration settings perform well, yet none of the settings tested are at least as good as SOL-IT$_{10}$(5,5) in all metrics.

In general, settings with an equal number of variable and fixed LCF scenarios and with at least seven iterations are observed to perform the best or very close to the best. Thus, for the remaining analysis, we stay with SOL-IT$_{10}$(5,5) as our preferred setting.

4.3 Additional Performance Information

In this section we cover a more detailed breakdown of the runtime, iteration performance, and the small performance analysis tests for SOL-IT$_{10}$(5,5), using solution data from the 100 problem instances. In Section 4.2 we calculate the average total runtime of SOL-IT$_{10}$(5,5) to be 200.7 seconds. The majority of this time, 145.2 seconds, is used for our original model, SLSF, to create the menus for each iteration. The remaining time is used largely to conduct the small performance analysis test, using a total of 42.3 seconds. Because the number of variable and fixed LCF scenarios is low, our LCF-FMS formulation requires very little time, with the total time for all 10 iterations being only 13.2 seconds.

Next, we are interested in the small performance test's ability to select good solutions. Our solution method SOL-IT iteratively finds new solutions, but the performance is not guaranteed to monotonically improve with each iteration. After each iteration, the small performance analysis test determines if the new solution should replace the current best solution. The decision to keep or discard the new solution is made correctly for 64.8% of the iterations, and the final solution returned is the 'best' solution, that is, the solution with the highest objective value in the large performance analysis test, in 23/100 problem instances. On average, the large performance analysis objective estimate of the returned solution is only 0.67% lower than the best solution, while the objective of the 'worst' found solution is on average 3.61%

Table 7: Performance averages and paired sample t-test p-values for settings using different numbers of iterations. SOL-IT is shortened to 'S.' for the row labels. Each row displays the average for each metric for the given setting, and resulting p-values from comparing each metric (excluding runtime) of SOL-IT$_{10}(5,5)$ (setting A) to the given setting. p-values above 0.99995 are displayed as 1, values below 0.00005 are displayed as 0, and values above the 0.05 threshold are bolded.

	Objective		**Unmat**		**Unhap**		**Profit ($)**		**Time (s)**
Setting B	Ave	p-val	Ave	p-val	Ave	p-val	Ave	p-val	Ave
S.$_{10}(5,5)$	164.14	–	0.687	–	0.130	–	81.45	–	200.7
S.$_1(5,5)$	162.13	0	1.215	0	0.137	**0.2822**	81.84	**0.9759**	31.4
S.$_1(10,10)$	162.28	0	1.279	0	0.155	0.0239	82.35	**1**	34.0
S.$_1(20,20)$	161.98	0	1.287	0	0.164	0.0023	82.35	**1**	47.5
S.$_1(50,50)$	162.11	0	1.276	0	0.161	0.0063	82.38	**1**	151.2
S.$_2(5,5)$	162.55	0	1.028	0	0.146	**0.0678**	81.53	**0.6849**	57.3
S.$_2(10,10)$	163.16	0	1.094	0	0.145	**0.0812**	82.22	**1**	63.2
S.$_2(20,20)$	162.93	0	1.073	0	0.165	0.0030	82.30	**1**	90.5
S.$_2(50,50)$	162.84	0	1.107	0	0.147	**0.1073**	82.11	**0.9999**	311.7
S.$_5(5,5)$	163.78	0.0169	0.785	0	0.126	**0.6660**	81.47	**0.6186**	120.1
S.$_5(10,10)$	163.75	**0.0548**	0.924	0	0.137	**0.2552**	82.04	**0.9999**	143.6
S.$_5(20,20)$	163.50	0.0018	0.906	0	0.154	0.0124	82.03	**0.9998**	220.5
S.$_5(50,50)$	163.68	0.0383	0.857	0.0001	0.146	**0.1330**	81.89	**0.9989**	816.0
S.$_7(5,5)$	163.78	0.0048	0.741	0.0013	0.130	**0.4907**	81.45	**0.4569**	152.8
S.$_7(10,10)$	164.18	**0.5793**	0.834	0.0011	0.125	**0.6774**	82.02	**1**	197.4
S.$_7(20,20)$	163.79	0.0369	0.852	0.0001	0.144	**0.1004**	82.02	**0.9999**	307.4
S.$_7(50,50)$	164.35	**0.8415**	0.794	0.0049	0.120	**0.8186**	81.86	**0.9994**	1125.4
S.$_{13}(5,5)$	163.83	0.0167	0.712	0.0433	0.135	**0.2737**	81.42	**0.3148**	246.3
S.$_{13}(10,10)$	164.40	**0.9152**	0.760	0.0370	0.139	**0.1944**	81.99	**1**	344.2
S.$_{13}(20,20)$	164.06	**0.3474**	0.735	**0.0788**	0.159	0.0104	81.89	**0.9997**	561.3

lower. Thus, while this method may not find the best solution, it still manages to find good solutions.

Lastly, we inspect the progression of the solution quality from iteration to iteration. The new current best solution's objective estimate (calculated from the large performance analysis) is higher than the previous iteration's stored best solution's objective by an average of 0.36% for the first three iterations and only by an average of 0.08% for the last three iterations. This result is also confirmed in our iteration experiment in Section 4.2, as there are notable improvements in performance proceeding from 1 to 2 to 5 iterations for each setting, while the performance differences between implementing 7, 10, or 13 iterations are much less pronounced.

4.4 Comparative Performance Analysis

In this section, we compare SOL-IT$_{10}(5,5)$ performance against other methods, which include SLSF (that optimizes menu composition given a fixed compensation scheme), our other non-iterative solution methods (see Figure 2), and a set of deterministic menu-generating methods: CLOS-1, LIK-1, CLOS-5, and LIK-5. Each of these deterministic methods solve a max-weight matching problem such that each

request is offered to one driver for CLOS-1 and LIK-1, and to five drivers for CLOS-5 and LIK-5. Further, each driver receives a menu of one request for CLOS-1 and LIK-1, and for CLOS-5 and LIK-5 each driver menu has five requests. For CLOS-1 and CLOS-5, the menu minimizes the sum of the time needed for the driver to drive to the request origin, that is, the wait time is minimized so requests will most likely be recommended to a close-by driver. LIK-1 and LIK-5 maximize the sum of the likelihoods a driver will accept each request, i.e. the p_{ij} values, which are fixed for these methods and derived using the same default compensation scheme as SLSF. We include CLOS-1 as well as LIK-1 to offer a comparison of our method against alternate methods that only recommend a single request to each driver (e.g., Lyft, Uber). CLOS-5 and LIK-5 provide examples of approaches that derive menus from a simplistic method as opposed to our method SOL-IT that selects menus and compensations to optimize performance. All create menus without a stochastic model for driver behavior (and are therefore deterministic), but the resulting menus are analysed using the same stochastic driver selection performance analysis described in Section 3.3. The results are displayed in Table 8. We refer to using SOL-DIR with A variable scenarios as SOL-DIR(A), using SOL-FM with A variable scenarios as SOL-FM(A), and using SOL-FMS with A variable scenarios and B fixed scenarios as SOL-FMS(A,B). Our experiment contains one setting configuration for each of these solution methods. The SLSF solutions are the menus used in the first iteration of all SOL-IT experiments, and are also used as the menus for SOL-FM and SOL-FMS.

Table 8: Performance averages and paired sample t-test p-values. Each row displays the given method's average metric value, and resulting p-values from comparing each performance metric of SOL-IT$_{10}$(5,5) (method A) to the given method, with p-values above 0.99995 being displayed as 1, values below 0.00005 displayed as 0, and values above the 0.05 threshold are bolded.

	Objective		**Unmat**		**Unhap**		**Profit ($)**		**Time (s)**
Method B	Ave	p-val	Ave	p-val	Ave	p-val	Ave	p-val	Ave
SOL-IT$_{10}$(5,5)	164.14	–	0.687	–	0.130	–	81.45	–	200.7
CLOS-1	115.57	0	7.081	0	0	**1**	56.13	0	0.017
LIK-1	130.98	0	4.718	0	0	**1**	64.75	0	0.016
CLOS-5	142.72	0	2.265	0	1.212	0	81.15	**0.0514**	0.016
LIK-5	155.99	0	1.360	0	0.592	0	83.46	**1**	0.015
SOL-DIR(10)	147.42	0	1.950	0	0.895	0	79.23	0	13.1
SOL-FM(10)	161.29	0	1.174	0	0.173	0.0082	81.73	**0.9220**	26.0
SOL-FMS(10,10)	162.26	0	1.129	0	0.148	**0.0694**	81.54	**0.6866**	27.1
SLSF	161.66	0	1.516	0	0.161	0	83.24	**1**	23.1

Table 8 shows that SOL-IT$_{10}$(5,5) has significantly better performance than all alternate methods in every solution performance metric with few exceptions. For the objective value and number of unmatched requests, SOL-IT$_{10}$(5,5) is statistically better than all alternate methods. The SOL-IT$_{10}$(5,5) objective average is 1.53% better than SLSF, and 11.34%, 1.76%, and 1.16% better than our initial solution methods SOL-DIR, SOL-FM, and SOL-FMS, respectively. Compared to the four deterministic methods, SOL-IT$_{10}$(5,5) has a much higher objective average, being 42.03% higher than CLOS-1, 25.32% higher than LIK-1, 15.01% higher than CLOS-5, and 5.22% higher than LIK-5. As expected, CLOS-1 and LIK-1 have fewer unhappy driver requests compared to SOL-IT$_{10}$(5,5); for these two methods, each request

is offered to exactly one driver, and an optimal recourse assignment will never have unhappy drivers. However, they perform poorly on objective and profit averages, primarily due to the large number of unmatched requests. LIK-5 has the highest profit average and lowest runtime out of all tested methods, but the objective average is low because there are more unmatched requests and unhappy driver requests. SLSF has a higher profit than SOL-IT because the SLSF compensation scheme is the minimum compensation for SOL-IT, meaning SOL-IT solutions can only keep or lower the portion of the fare kept by the platform for each request. Thus, SLSF may have a higher profit over a single time period. However, in the long term, the higher number of unhappy drivers and unmatched requests (meaning the rider must wait until a later time period to be matched or leave the platform) observed when using SLSF may result in losses in the number or frequency of drivers and riders, causing loss of profit.

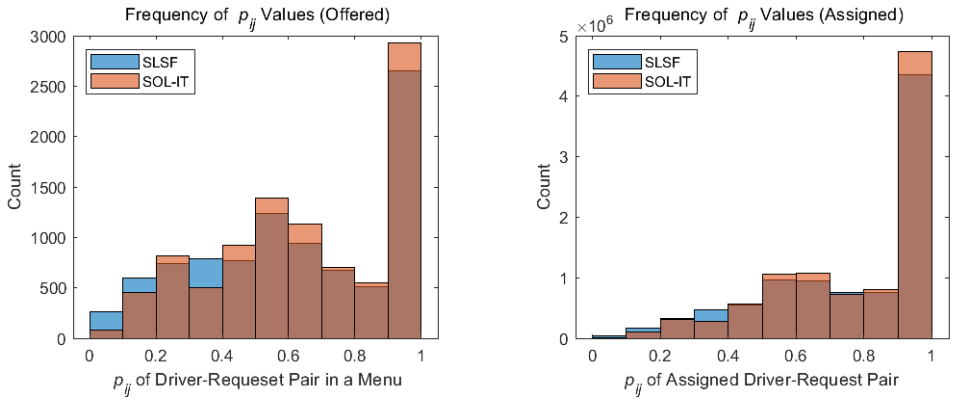
4.5 Decision Variable Insights

In this section we provide insights into how a platform should create menus and incentivize offers. To inform these insights, we explore the decision variable values from the 100 problem instances produced by SOL-IT$_{10}$(5,5). We use SLSF solutions as a point of comparison to highlight the differences between optimizing incentives and menus (SOL-IT) as opposed to a *menu-only* optimization with fixed compensation (SLSF). We begin with general observations about the solutions, followed by more specific observations concerning the compensation decisions.

First, we explore the general contents of drivers' menus. The average menu size for SLSF solutions is 4.59 out of the maximum five, while the average menu size of the SOL-IT$_{10}$(5,5) menus is 4.75. This difference is statistically significant under a paired sample t-test. Thus, a platform with personalized incentives creates larger menus, on average, than a fixed compensation policy. Each iteration's menu contains approximately 73.0% of the driver-request pairs in the initial SLSF menu. In addition, with each iteration, approximately 72% of the driver-request pairs from the previous iteration's menu set remain in the new menu set. The overlap for menus in later iterations is about 2% higher than those of earlier iterations. In comparison, as discussed in Section 4.3, objective value improvement quickly dwindles and is close to zero by the 10th iteration, indicating a lack of merit in additional iterations in pursuit of full solution convergence. Some *core* driver-request pairs are in all 10 iterations' menus, but these pairs make up on average only 34.15% of a given menu. These common elements consist of a select number of driver-request pairs making good menu items (requests in which the platform makes a profit and drivers are willing to accept). For example, the average p_{ij} (under the default minimum compensation) of the core driver-request pairs in all 10 iteration's menus is 0.776, while the average across all requests in the initial SLSF menu is only 0.630. The driver-request pairs kept for all iterations have a high average p_{ij} even at the minimum compensation, meaning the platform can make a sizeable profit and have decent driver participation, or they can add incentives to increase the driver participation even further. Either way, having a high initial p_{ij} is attractive to the platform, meaning requests more efficient for the driver (less extra driving time) are more likely to appear in a menu for both SLSF and SOL-IT.

In Figure 3 we compare the distribution of p_{ij} values in the SOL-IT returned solutions with the p_{ij} values of the SLSF solutions (which are under the minimum compensation scheme). Recall that $\ddot{p}_{ij}$

is the willingness decision variable returned by SOL-IT, and acceptance probability p_{ij} is calculated as $p_{ij} = \min\{\ddot{p}_{ij}, 1\}$. Figure 3a displays the frequency of different p_{ij} values of driver-request pairs offered in the menus of the SLSF solutions and the returned SOL-IT solutions, and Figure 3b displays the p_{ij} frequencies of the driver-request pairs assigned in each test scenario of the performance analysis for the SLSF and SOL-IT solutions. For example, a driver-request pair assigned in 1,000 scenarios out of the 5,000 test scenarios, then the corresponding p_{ij} value is represented as 1,000 entries on the histogram. For both SLSF and SOL-IT, roughly 30% of the offered driver-request pairs have a p_{ij} value of at least 0.9, but many pairs are in the mid p_{ij} range and even some in the low p_{ij} range. Meanwhile, the driver-request pairs assigned must also be accepted by the driver, so the number of p_{ij} values in the low range is very small, and almost half of pairs have a p_{ij} value of at least 0.9. SOL-IT has noticeably fewer offered requests with $p_{ij} < 0.2$ compared to SLSF. The reason there are almost no SOL-IT menu items with $p_{ij} < 0.1$ is that with a premium value of 0.75 and a total of 10 training LCF scenarios (five variable and five fixed), if $\ddot{p}_{ij}$ is below $0.1\bar{3}$ then driver j cannot accept request i in any of the training scenarios. However, the total number of SOL-IT menu items with $p_{ij} < 0.2$ being much lower suggests the solution does not highly benefit from these types of requests. Having low p_{ij} requests in menus takes up a menu slot and is only accepted on rare occasions. The probability the request is accepted is already low, and the chances of being assigned are even lower: for SOL-IT driver-request pairs with $p_{ij} < 0.2$, the pair is assigned in only an average of 25.44% of the test scenarios where the driver accepted the request. Instead of spending resources on these 'fringe' menu items, through compensation manipulation, SOL-IT solutions are able to shift the p_{ij} distribution to slightly higher values with a mean of $p_{ij} = 0.661$ compared to a mean of $p_{ij} = 0.630$ for the fixed compensation approach.

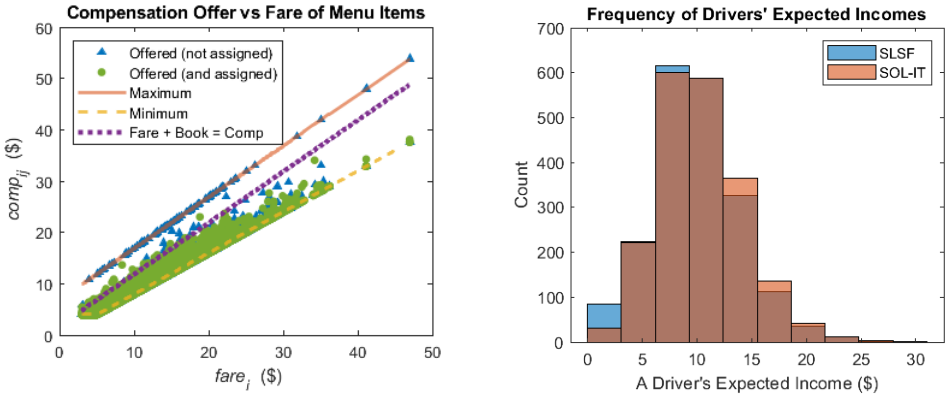


(a) (b)

Figure 3: Frequencies of p_{ij} values for SLSF and SOL-IT: (a) is the p_{ij} of driver-request pairs in menus (Offered), and (b) is the p_{ij} of driver-request pairs assigned in the test scenarios (Assigned).

Next, we analyze compensation decisions. We start by comparing, for each driver-request pair in a menu, the portion of the fair offered to the driver (i.e. $comp^*_{ij}/fare_i$); see Figure 4a. Most items offered in menus are offered at their minimum compensation (i.e., 71.4% of items offered have the minimum compensation of 80% of the fare or \$4 for requests with fare < \$5). Further, a high percentage of the menu items, 93.3%, have compensation offers lower than the fare so the platform would make at least

the $1.85 booking fee. Only 1.47% of menu items are above the profit equal zero line, and only 25.9% of them are assigned in one or more of the fixed or variable training scenarios. To understand the impact of allowing driver compensations to be high enough that the platform can lose money on an individual request, we conducted a separate experiment with lower compensation bounds. For this experiment we use SOL-IT$_{10}$(5,5) on the 100 problem instances with all the same input parameters except that the maximum compensation is the fare plus the booking fee, so the platform will always at least break even on each request assigned. The resulting average profit is only $0.05 higher than the solutions using our original maximum compensation, but the objective average, unmatched request average, and unhappy driver averages are all worse (averages are 163.88, 0.741, and 0.139, respectively). Therefore, while our compensation bound choices can have small losses individually, the improved rider and driver experiences would likely increase repeat ridership and driver log-ins, which is good for the platform over time.



(a) (b)

Figure 4: Graph (a) shows the compensation offers in a menu compared to the fare of the request. The solid orange and dashed yellow curves represent the maximum and minimum compensation allowed, respectively, and the purple dotted line is the profit = 0 line. Driver-request pairs in a menu not assigned in any of the fixed or variable LCF scenarios are shown as blue triangles, while pairs assigned in at least one fixed or variable LCF scenario are green circles. Graph (b) is a histogram of the expected income of all of the drivers for SLSF and SOL-IT, calculated as a weighted average of their individual income in each test scenario in the large performance analysis.

Next, we examine the impact of SOL-IT's policy on expected incomes for drivers. Figure 4b depicts a histogram of the expected incomes for all of the drivers in the 100 problem instances for SLSF and SOL-IT$_{10}$(5,5). The number of drivers in the lowest expected income brackets is lower for SOL-IT compared to SLSF, and the number of drivers in the middle income brackets is higher. This shift in expected driver income distribution results in each driver earning an average of $10.33, compared to drivers in a system without added incentives (SLSF), who can expect an average of only $9.92 each. Across the 20 drivers, this is an increase of $8.20 in total expected driver income, while the platform profit only decreases by $1.79, meaning 78.2% of this increased income comes from the additional revenue obtained from matching more requests and more efficient requests than the fixed compensation case of SLSF.

In Figure 5, we compare the compensation level to the final (Figure 5a) and original (Figure 5b) driver willingness levels of accepting the requests in their menu. The compensation level is measured as the

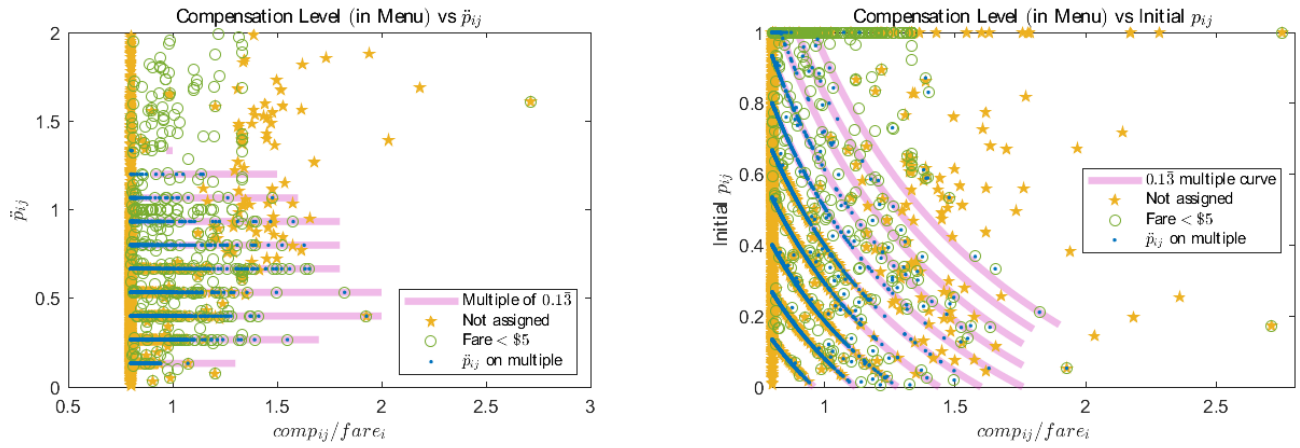
portion of the fare offered (i.e. $comp^*_{ij}/fare_i$), the final driver willingness is the $\ddot{p}_{ij}$ values in the returned solution, and the initial willingness is the p_{ij} values observed under the first iteration's minimum compensation scheme. Figure 5a is cropped to zoom in on driver-request pairs with $\ddot{p}_{ij}$ values below 2. In Figure 5a, 46.7% of compensation offers above 80% of the fare fall on one of the plotted pink lines. These lines represent the horizontal lines $\ddot{p}_{ij} = 0.1\bar{3}k$ for $k = 1, 2, ..., 10$, which occur because $0.1\bar{3}$ is the minimum $\ddot{p}_{ij}$ value needed for driver j to accept request i in exactly one fixed or variable LCF scenario, and $0.2\bar{6}$ is the minimum for the driver to accept request i in exactly two LCF scenarios, and so on. Once the solver has determined the optimal set of driver responses for the set of variable LCF scenarios (as the fixed scenarios are already decided), the drivers receive the minimum offer guaranteeing that level of participation. Raising the offer further only cuts into platform profits. The offers with compensation level >80% not on one of the multiple-of-$0.1\bar{3}$ lines falls into one of two categories. First, if the fare is under \$5 then the minimum compensation is above 80% of the fare, so if no incentives are added these offers do not lie on a multiple-of-$0.1\bar{3}$ line. The compensation can also yield a $\ddot{p}_{ij}$ not on a $0.1\bar{3}$ multiple if the driver-request pair is not assigned in any of the 10 (fixed/variable) LCF training scenarios, as driver j does not have a chance to accept request i in any LCF scenarios if the pair is never assigned.

For the pairs with $\ddot{p}_{ij}$ on a $0.1\bar{3}k$ line in Figure 5a, most of the higher compensation offers are for k values between 3 and 7 (i.e. the driver accepts in 30%-70% of training scenarios). The compensations for $k = 1$, $k = 9$, and $k = 10$ are almost all near the minimum 80%. These patterns are consistent with our findings that SOL-IT prioritizes efficient requests with potential to benefit the platform the most. The solver does not spend the resources to offer high incentive levels on a request just for the driver to accept the request in one or two LCF training scenarios out of 10 ($\ddot{p}_{ij} = 0.1\bar{3}$ and $\ddot{p}_{ij} = 0.2\bar{6}$). Instead, the higher incentives are given to promising driver-request pairs that yield a strong chance of acceptance so they can be used in several scenario assignments. The lower offers for $k = 9$ and $k = 10$ are because the optimal assignment rarely needs a given driver-request pair to be assigned in every or almost every LCF training scenario, so there are only a few incentive offers in this $\ddot{p}_{ij}$ range. Further, the driver-request pairs that the solver would assign in 9 or 10 training scenarios are typically naturally efficient, having low extra driving time for the driver. Thus, extra incentives are not needed for high participation. This is confirmed shortly in our discussion of Figure 5b.

While Figure 5a shows the relationship between the final willingness and the compensation level, Figure 5b provides insight into how and where the driver willingness is changed by the addition of incentives. Like in Figure 5a, a large portion of the menu items fall on one of 10 curves, also plotted in pink. These curves represent the portion of the fare needed to be offered to increase a given initial p_{ij} value to a solution $\ddot{p}_{ij}$ value equal to $0.1\bar{3}k$ for $k = 1, 2, ..., 10$. These curves are given by the equation,

$$p_{ij} = \frac{0.8(0.1\bar{3}k + 0.\bar{6})}{comp_{ij}/fare_i} - 0.\bar{6}$$

for $k = 1, 2, ..., 10$, derived from Equation (1). These curves assume the minimum compensation is 80% of the fare, so driver-request pairs with $fare_i <$ \$5 do not fall on these curves as the minimum compensation is higher than 80% of the fare. This is why all of the driver-request pairs in Figure 5b with a multiple of $0.1\bar{3}$ $\ddot{p}_{ij}$ value (i.e. all of the blue dots) that don't fall on one of the pink curves also have a fare below



(a) (b)

Figure 5: Graphs comparing portion of the fare offered in a menu to the driver's willingness to accept the request. Graph (a) shows the driver's willingness to accept in the returned solution, captured as $\ddot{p}_{ij}$, while (b) shows the driver's willingness to accept the request under the minimum compensation scheme used as the compensation initialization. Driver-request pairs with a fare below \$5 are plotted as a green circle, and pairs offered but not assigned in any of the fixed or variable training scenarios are plotted as a yellow star. If the final $\ddot{p}_{ij}$ value is a multiple of $0.1\bar{3}$, the pair is plotted as a blue dot. Any pairs fitting multiple categories are plotted with each symbol, and pairs fitting none of these categories are omitted as all fall on the $comp_{ij}/fare_i = 0.8$ line.

\$5, and a green circle is plotted around each of these blue dots.

Out of the driver-request pairs on the top three pink lines (i.e. the driver accepts the request in ≥ 8 LCF training scenarios), almost all have a relatively high initial p_{ij} value. This supports our earlier hypothesis that the highest participation rate pairs are naturally efficient for the driver. Figure 5b also shows how a driver's willingness to accept is increased by the compensations. For example, having $p_{ij} \leq 0.2$ initially means the driver-request pair could only be accepted/assigned in up to one of the 10 LCF training scenarios at the minimum compensation rate, but several of these pairs have the compensation increased enough for the driver to be willing to accept in two, three, or even four training scenarios. These can be identified in Figure 5b as the driver-request pairs with $p_{ij} \leq 0.2$ plotted on the 2nd, 3rd, and 4th pink lines from the bottom. A few pairs with initial $p_{ij} \leq 0.2$ are accepted in even more scenarios, but these are much less common, and none of them are accepted in all 10 training scenarios. This limit in increasing driver participation occurs because only so much compensation can be offered while still benefiting the platform. For example, to raise driver willingness from $p_{ij} = 0.2$ to be high enough to be accepted in all training scenarios, the driver must be offered almost two times the fare, which is unprofitable especially because the fare is >\$5 for these pairs on the pink lines.

5 Conclusions and Future Directions

To provide an enhanced driver experience and a means for the platform to efficiently match drivers and requests, we create and solve a stochastic optimization model to produce individualized menus of re-

quests with corresponding individualized incentive levels for each request. We capitalize on a variable property along with McCormick constraints, enabling our model to be formulated as a linear integer program. This is a complex multi-stage problem because of stochastic driver selections, with a driver's probability to accept a request affected by the offered compensation. Drawing inspiration from the Sample Average Approximation (SAA) method, our model captures this endogeneity by modeling different scenarios of possible driver responses as variables to approximate expected solution performance. To counterbalance overfitting of decisions based on these variable scenarios, we impose a premium, requiring the platform to offer drivers a higher compensation than is needed for the solution's desired participation levels. A series of solution methods improve solution quality, culminating in our SOL-IT method. First, the formulation is decomposed into two steps: we produce a menu set that considers compensation as given, using the SLSF formulation. These menus are then fixed, and fed into a formulation to determine optimal compensation offers. The menu decision subproblem can use exogenous (fixed) driver scenarios, while the compensation decision subproblem is hybridized, using both fixed and variable driver selection scenarios, to further prevent overfitting. Lastly, we iterate between the two subproblems, and implement small performance tests to select the best solution. The Chicago Regional Transportation network provides experiment rider and driver trips influenced by real-world data.

We conduct a sequence of experiments, confirming a compensation premium improves solution performance, establishing that several different variable and fixed scenario sample sizes yield well-performing solutions, and providing a sample size recommendation. Iterating is demonstrated to benefit solution quality, with 10 iterations being sufficiently high to gain these benefits with reasonable computational time. Our small performance analysis test is shown to be an effective method to select the best solution, in which the objective value of the selected solution is on average only 0.67% lower than the actual best solution. A comparative analysis juxtaposes the performance of SOL-IT with alternative deterministic approaches, a fixed-compensation menu-optimizing formulation with stochastic driver responses (i.e. the solution from SLSF), and other variants of our solution method (see Figure 2). SOL-IT with our recommended settings has statistically better objective values, matches the most requests, and has the lowest occurrence of drivers accepting a request(s) but not receiving a match (excluding two deterministic approaches that do not allow accepting drivers to not be matched, but perform poorly otherwise).

We perform an in-depth analysis of our solution decision variables for additional insights, finding that SOL-IT strategically incentivizes menu items to efficiently match requests. Further, a platform with personalized incentives creates larger menus, on average, than a fixed compensation policy. Also, 30% of the menu items are different than the menus created using the stochastic fixed-compensation menu-optimization approach, meaning the iterative process and addition of incentives significantly change the menus, not just the compensation. Simultaneously changing incentives and menu composition results in higher driver participation: on average, a driver is willing to accept a given request in the fixed-compensation (no incentives) solution's menu set with probability 0.630, and this average probability increases to 0.661 for the returned SOL-IT menu (with the optimized menus and incentives). On average, the 20 available drivers receive a total \$8.20 more compensation under personalized incentives (a 4.1% increase from fixed compensation) and the platform achieves a \$6.41 revenue increase due to

a higher match rate. Only 21.8% of the drivers' pay increase comes out of platform profit (which decreases 2.2% from fixed compensation). The driver-request pairs receiving the highest incentives have mid-range willingness levels after the incentive is added (i.e. driver accepting in 30-70% of training scenarios), making them dependable assignment options for the platform. Further, the platform spends less resources to incentivize driver-request pairs with lower participation rates: fewer menu pairs with a low willingness rate occur in the SOL-IT solutions than in the no-incentive solutions, and the SOL-IT incentive offers for these pairs are lower than those of the $30-70\%$ participation pairs.

As this is the first to personalize both menus and incentives in ridesharing or crowdsourced delivery, a number of future directions can be taken. Finding methods or heuristics to reduce runtime could allow SOL-IT to be implemented in practice with reasonably short decision epochs. Improvements are warranted for SLSF (for example using fewer training scenarios or simplifying the formulation) or the performance analysis run each iteration (finding faster and possibly more accurate alternative methods), as these two components take up the majority of SOL-IT's runtime. New models could consider and quantify increased equity for customer and driver experiences, for example maximizing the minimum service level for customers or requiring drivers in similar locations to receive the same incentive levels. A multiple period and dynamic model could be developed to capture future time periods with newly arriving drivers and requests. Additional considerations could enforce/encourage a higher level of service, for example prioritizing matching requests/drivers unmatched in the previous time period.

Acknowledgements

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

References

Alnaggar, Aliaa, Gzara, Fatma, & Bookbinder, James H. 2019. Crowdsourced Delivery: A Review of Platforms and Academic Literature. *Omega*, **98**, 102139. DOI: 10.1016/j.omega.2019.102139.

Asghari, Mohammad, & Shahabi, Cyrus. 2018. Adapt-pricing: a dynamic and predictive technique for pricing to maximize revenue in ridesharing platforms. *Pages 189–198 of: Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. DOI: 10.1145/3274895.3274928.

Ausseil, Rosemonde, Pazour, Jennifer, & Ulmer, Marlin. 2021. *Supplier Menus for Dynamic Matching in Peer-to-Peer Transportation Platforms*. Working paper, https://www.researchgate.net/publication/350016164_Supplier_Menus_for_Dynamic_Matching_in_Peer-to-Peer_Transportation_Platforms (Retrieved on March 29 2021).

Bai, Jiaru, So, Kut C, Tang, Christopher S, Chen, Xiqun, & Wang, Hai. 2019. Coordinating supply and demand on an on-demand service platform with impatient customers. *Manufacturing & Service Operations Management*, **21**(3), 556–570. DOI:10.2139/ssrn.2831794.

Banerjee, Siddhartha, Riquelme, Carlos, & Johari, Ramesh. 2015. Pricing in ride-share platforms: A queueing-theoretic approach. *Available at SSRN 2568258*. DOI: 10.2139/ssrn.2568258.

Barbosa, Miguel Moreira da Silva Lima. 2019. *A data-driven compensation scheme for last-mile delivery with crowdsourcing*. M.Phil. thesis, University of Porto. https://repositorio-aberto.up.pt/bitstream/10216/124212/2/367287.pdf (Retrieved on May 14 2020).

Bimpikis, Kostas, Candogan, Ozan, & Saban, Daniela. 2019. Spatial pricing in ride-sharing networks. *Operations Research*, **67**(3), 744–769. DOI: 10.1287/opre.2018.1800.

Cachon, Gerard P, Daniels, Kaitlin M, & Lobel, Ruben. 2017. The role of surge pricing on a service platform with self-scheduling capacity. *Manufacturing & Service Operations Management*, **19**(3), 368–384. DOI: 10.1287/msom.2017.0618.

Castillo, Juan Camilo, Knoepfle, Daniel T, & Weyl, E Glen. 2018. Surge Pricing Solves the Wild Goose Chase. *Available at SSRN 2890666*. DOI: 10.2139/ssrn.2890666.

Chen, Mengjing, Shen, Weiran, Tang, Pingzhong, & Zuo, Song. 2018a. Optimal vehicle dispatching for ride-sharing platforms via dynamic pricing. *Pages 51–52 of: Companion Proceedings of the The Web Conference 2018*. DOI: 10.1145/3184558.3186924.

Chen, Wenyi, Mes, Martijn, & Schutten, Marco. 2018b. Multi-hop driver-parcel matching problem with time windows. *Flexible services and manufacturing journal*, **30**(3), 517–553. DOI: 10.1007/s10696-016-9273-3.

Chen, Xiqun Michael, Zheng, Hongyu, Ke, Jintao, & Yang, Hai. 2020. Dynamic optimization strategies for on-demand ride services platform: Surge pricing, commission rate, and incentives. *Transportation Research Part B: Methodological*, **138**, 23–45. DOI: 10.1016/j.trb.2020.05.005.

Chicago Area Transportation Study (CATS). 2016. *Chicago Regional Network*. https://github.com/bstabler/TransportationNetworks/tree/master/chicago-regional (Retrieved on August 1, 2019).

de Ruijter, Arjan, Cats, Oded, Alonso-Mora, Javier, & Hoogendoorn, Serge. 2020. Ride-sharing efficiency and level of service under alternative demand, behavioral and pricing settings. *In: Transportation Research Board 2020 Annual Meeting*. https://scholar.google.com/scholar_lookup?title=Ride-sharing%20efficiency%20and%20level%20of% 20service%20under%20alternative%20demand%2C%20behavioral%20and%20pricing%20settings& publication=_year=2020&author=A.%20de%20Ruijter&author=C.%20Oded&author=A.-M.%20Javier&author= H.%20Serge (Retrieved on April 30, 2020).

Di Febbraro, A, Gattorna, E, & Sacco, N. 2013. Optimization of dynamic ridesharing systems. *Transportation research record*, **2359**(1), 44–50. DOI: 10.3141/2359-06.

EstimateFares.com. 2019. *Chicago Lyft Rates*. https://estimatefares.com/rates/chicago (Retrieved on August 1, 2019).

Furuhata, Masabumi, Dessouky, Maged, Ordóñez, Fernando, Brunet, Marc-Etienne, Wang, Xiaoqing, & Koenig, Sven. 2013. Ridesharing: The State-of-the-Art and Future Directions. *Transportation Research Part B: Methodological*, **57**, 28–46. DOI: 10.1016/j.trb.2013.08.012.

Gdowska, Katarzyna, Viana, Ana, & Pedroso, João Pedro. 2018. Stochastic last-mile delivery with crowdshipping. *Transportation Research Procedia*, **30**, 90–100. DOI: 10.1016/j.trpro.2018.09.011.

Helling, Brett. 2019. *Uber Fees: How Much Does Uber Pay, Actually? (With Case Studies)*. Ridester. https://www.ridester.com/uber-fees/ (Retrieved on April 30, 2020).

Hong, Huiting, Li, Xin, He, Daqing, Zhang, Yiwei, & Wang, Mingzhong. 2019. Crowdsourcing Incentives for Multi-Hop Urban Parcel Delivery Network. *IEEE Access*, **7**, 26268–26277. DOI: 10.1109/ACCESS.2019.2896912.

Horner, Hannah, Pazour, Jennifer, & Mitchell, John E. 2021. Optimizing Driver Menus Under Stochastic Selection Behavior for Ridesharing and Crowdsourced Delivery. *Transportation Research Part E: Logistics and Transportation Review*, **153**. DOI: 10.1016/j.tre.2021.102419.

Hu, Ming, & Zhou, Yun. 2019. Price, wage and fixed commission in on-demand matching. *Available at SSRN 2949513*. DOI: 10.2139/ssrn.2949513.

Jacob, Jagan, & Roet-Green, Ricky. 2018. *Ride solo or pool: designing price-service menus for a ride-sharing platform*. DOI: 10.2139/ssrn.3008136.

JC. 2019. *Have You Been Sidelined by Uber*. Ridester. https://www.ridester.com/have-you-been-sidelined-by-uber/ (Retrieved on April 30, 2020).

Jiang, Jingjing. 2019. *More Americans are using ride-hailing apps*. Pew Research Center. https://www.pewresearch.org/fact-tank/2019/01/04/more-americans-are-using-ride-hailing-apps/ (Retrieved on March 8, 2021).

Ke, Jintao, Yang, Hai, Li, Xinwei, Wang, Hai, & Ye, Jieping. 2020. Pricing and equilibrium in on-demand ride-pooling markets. *Transportation Research Part B: Methodological*, **139**, 411–431. DOI: 10.1016/j.trb.2020.07.001.

Kleywegt, Anton J, Shapiro, Alexander, & Homem-de Mello, Tito. 2002. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, **12**(2), 479–502. DOI: 10.1137/S1052623499363220.

Le, Tho V, Stathopoulos, Amanda, Van Woensel, Tom, & Ukkusuri, Satish V. 2019. Supply, demand, operations, and management of crowd-shipping services: A review and empirical evidence. *Transportation Research Part C: Emerging Technologies*, **103**, 83–103. DOI: 10.1016/j.trc.2019.03.023.

Le, Tho V, Ukkusuri, Satish V, Xue, Jiawei, & Van Woensel, Tom. 2021. Designing pricing and compensation schemes by integrating matching and routing models for crowd-shipping systems. *Transportation Research Part E: Logistics and Transportation Review*, **149**, 102209. DOI: 10.1016/j.tre.2020.102209.

Lei, Chao, Jiang, Zhoutong, & Ouyang, Yanfeng. 2019. Path-based dynamic pricing for vehicle allocation in ridesharing systems with fully compliant drivers. *Transportation Research Part B: Methodological*, **132**, 60–75. DOI: 10.1016/j.trb.2019.01.017.

Li, Minne, Qin, Zhiwei, Jiao, Yan, Yang, Yaodong, Wang, Jun, Wang, Chenxi, Wu, Guobin, & Ye, Jieping. 2019a. Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning. *Pages 983–994 of: The World Wide Web Conference*. DOI: 10.1145/3308558.3313433.

Li, Yafei, Wan, Ji, Chen, Rui, Xu, Jianliang, Fu, Xiaoyi, Gu, Hongyan, Lv, Pei, & Xu, Mingliang. 2019b. Top-k Vehicle Matching in Social Ridesharing: A Price-aware Approach. *IEEE Transactions on Knowledge and Data Engineering*. DOI: 10.1109/TKDE.2019.2937031.

Liu, Yang, & Li, Yuanyuan. 2017. Pricing scheme design of ridesharing program in morning commute problem. *Transportation Research Part C: Emerging Technologies*, **79**, 156–177. DOI: 10.1016/j.trc.2017.02.020.

Luo, Qi, & Saigal, Romesh. 2017. Dynamic pricing for on-demand ride-sharing: A continuous approach. *Available at SSRN 3056498*. DOI: 10.2139/ssrn.3056498.

Lyft. 2021. *Personal Power Zones*. https://help.lyft.com/hc/ko/articles/115012926807-Personal-Power-Zones (Retrieved March 9, 2021).

Ma, Hongyao, Fang, Fei, & Parkes, David C. 2020a. Spatio-temporal pricing for ridesharing platforms. *ACM SIGecom Exchanges*, **18**(2), 53–57. DOI: 10.1145/3328526.3329556.

Ma, Jie, Xu, Min, Meng, Qiang, & Cheng, Lin. 2020b. Ridesharing user equilibrium problem under OD-based surge pricing strategy. *Transportation Research Part B: Methodological*, **134**, 1–24. DOI: 10.1016/j.trb.2020.02.001.

Marshall, Aarian. 2020. *Uber Changes Its Rules, and Drivers Adjust Their Strategies*. Wired. https://www.wired.com/story/uber-changes-rules-drivers-adjust-strategies/. (Retrieved on April 27, 2021).

Masoud, Neda, & Jayakrishnan, R. 2017. A decomposition algorithm to solve the multi-hop Peer-to-Peer ride-matching problem. *Transportation Research Part B: Methodological*, **99**, 1–29. DOI: 10.1016/j.trb.2017.01.004.

Mofidi, Seyed Shahab, & Pazour, Jennifer A. 2019. When is it beneficial to provide freelance suppliers with choice? A hierarchical approach for peer-to-peer logistics platforms. *Transportation Research Part B: Methodological*, **126**, 1–23. DOI: 10.1016/j.trb.2019.05.008.

Mordor Intelligence. 2020. *Ridesharing Market - Growth, Trends, COVID-19 Impact, And Forecasts (2021 - 2026)*. https://www.mordorintelligence.com/industry-reports/ridesharing-market (Retrieved on March 8, 2021).

Najmi, Ali, Rey, David, & Rashidi, Taha H. 2017. Novel dynamic formulations for real-time ride-sharing systems. *Transportation Research Part E: Logistics and Transportation Review*, **108**, 122–140. DOI: 10.1016/j.tre.2017.10.009.

Nourinejad, Mehdi, & Ramezani, Mohsen. 2019. Ride-Sourcing modeling and pricing in non-equilibrium two-sided markets. *Transportation Research Part B: Methodological*, **132**, 340–357. DOI: 10.1016/j.trb.2019.05.019.

Porterfield, Carlie. 2021. *Uber Will Reclassify U.K. Drivers As Workers, A First For The Ride-Share Giant*. Forbes. https://www.forbes.com/sites/carlieporterfield/2021/03/16/uber-will-reclassify-uk-drivers-as- workers-a-first-for-the-ride-share-giant/?sh=1da04c313c97 (Retrieved on April 5, 2021).

Qin, Zhiwei, Tang, Xiaocheng, Jiao, Yan, Zhang, Fan, Xu, Zhe, Zhu, Hongtu, & Ye, Jieping. 2020. Ride-Hailing Order Dispatching at DiDi via Reinforcement Learning. *INFORMS Journal on Applied Analytics*, **50**(5), 272–286. DOI:10.1287/inte.2020.1047.

Rasulkhani, Saeid, & Chow, Joseph YJ. 2019. Route-cost-assignment with joint user and operator behavior as a many-to-one stable matching assignment game. *Transportation Research Part B: Methodological*, **124**, 60–81. DOI: 10.1016/j.trb.2019.04.008.

Stiglic, Mitja, Agatz, Niels, Savelsbergh, Martin, & Gradisar, Mirko. 2015. The benefits of meeting points in ride-sharing systems. *Transportation Research Part B: Methodological*, **82**, 36–53. DOI: 10.2139/ssrn.2567274.

Sun, Luoyi, Teunter, Ruud H, Babai, M Zied, & Hua, Guowei. 2019. Optimal pricing for ride-sourcing platforms. *European Journal of Operational Research*, **278**(3), 783–795. DOI: 10.1016/j.ejor.2019.04.044.

Tafreshian, Amirmahdi, Masoud, Neda, & Yin, Yafeng. 2020. Frontiers in Service Science: Ride Matching for Peer-to-Peer Ride Sharing: A Review and Future Directions. *Service Science*. DOI: 10.1287/serv.2020.0258.

Uber. 2020. *Uber Ride Options*. https://www.uber.com/us/en/ride/ride-options/ (Retrieved on March 8, 2021).

Wang, Hai, & Yang, Hai. 2019. Ridesourcing systems: A framework and review. *Transportation Research Part B: Methodological*, **129**, 122–155. DOI: 10.1016/j.trb.2019.07.009.

Wang, Yuan, Zhang, Dongxiang, Liu, Qing, Shen, Fumin, & Lee, Loo Hay. 2016. Towards enhancing the last-mile delivery: An effective crowd-tasking model with scalable solutions. *Transportation Research Part E: Logistics and Transportation Review*, **93**, 279–293. DOI: 10.1016/j.tre.2016.06.002.

Wu, Yongzhong, Chen, Xiangying, & Ma, Jingwen. 2018. Modeling Passengers' Choice in Ride-Hailing Service with Dedicated-Ride Option and Ride-Sharing Option. *Pages 94–98 of: Proceedings of the 4th International Conference on Industrial and Business Engineering*. DOI: 10.1145/3288155.3288199.

Yan, Chiwei, Zhu, Helin, Korolko, Nikita, & Woodard, Dawn. 2020. Dynamic pricing and matching in ride-hailing platforms. *Naval Research Logistics (NRL)*, **67**(8), 705–724. DOI: 10.1002/nav.21872.

Zebra. 2018. *Reinventing the Supply Chain: the Future of Fulfillment Vision Study*. https://www.zebra.com/content/dam/zebra_new_ia/en-us/solutions-verticals/vertical-solutions/retail/vision-study/fulfillment-vision-study-report-en-us.pdf (Retrieved on May 18, 2021).

Zha, Liteng, Yin, Yafeng, & Du, Yuchuan. 2018. Surge pricing and labor supply in the ride-sourcing market. *Transportation Research Part B: Methodological*, **117**, 708–722. DOI: 10.1016/j.trb.2017.09.010.

A Algorithm Overview of SOL-IT

Algorithm A.1 outlines the process used to execute SOL-IT and how the returned solution is selected.

Algorithm A.1 SOL-IT

Generate fixed SLSF scenarios under default compensation $fixcomp^{(1)}$.
Run SLSF with $\boldsymbol{fixcomp}^{(1)} \longrightarrow$ get menu $\boldsymbol{x}^{(0)}$.
For iteration 1, set $\boldsymbol{comp}^{(0)} := \boldsymbol{fixcomp}^{(1)}$.
Run small performance analysis test on $[\boldsymbol{x}^{(0)}, \boldsymbol{comp}^{(0)}]$ yielding objective estimate $obj^{(0)}$.
Initialize $iter_{best} = 0$ and $obj_{best} = obj^{(0)}$.
Generate fixed LCF scenarios under $fixcomp^{(1)}$.
For iteration 1, set $\boldsymbol{x}^{(1)} := \boldsymbol{x}^{(0)}$.
Run LCF-FMS with $\boldsymbol{x}^{(1)} \longrightarrow$ get $\boldsymbol{comp}^{(1)}$.
Small performance analysis of $[\boldsymbol{x}^{(1)}, \boldsymbol{comp}^{(1)}] \longrightarrow$ get $obj^{(1)}$.
if $obj^{(1)} > obj_{best}$ **then**
    $obj_{best} \longleftarrow obj^{(1)}$, $iter_{best} \longleftarrow 1$.
end if
for $k = 2 : iterations$ **do**
    Use $\boldsymbol{comp}^{(k-1)}$ to update fixed compensation$\longrightarrow$ get $fixcomp^{(k)}$.
    Generate fixed SLSF scenarios under $fixcomp^{(k-1)}$.
    Run SLSF $\longrightarrow$ get menu $\boldsymbol{x}^{(k)}$.
    Generate fixed LCF scenarios under $fixcomp^{(k-1)}$.
    Run LCF-FMS with $\boldsymbol{x}^{(k)} \longrightarrow$ get $\boldsymbol{comp}^{(k)}$.
    Small performance analysis of $[\boldsymbol{x}^{(k)}, \boldsymbol{comp}^{(k)}] \longrightarrow$ get $obj^{(k)}$.
    if $obj^{(k)} > obj_{best}$ **then**
        $obj_{best} \longleftarrow obj^{(k)}$, $iter_{best} \longleftarrow k$.
    end if
end for
Return $\boldsymbol{x}^{(iter_{best})}$ and $\boldsymbol{comp}^{(iter_{best})}$.

B Single-Level Stochastic Formulation (SLSF)

Our Single-Level Stochastic Formulation (SLSF) presented in Horner *et al.* (2021) is given by (B.1)-(B.9), which takes as input a fixed compensation scheme $\boldsymbol{fixcomp}$. With compensation fixed, driver willingness to accept requests is not endogenous so training scenarios are fixed instead of variable and generated beforehand. An *SLSF training scenario* is denoted as $\bar{s}$, and the sample set of fixed training scenarios is denoted as $\bar{\Omega}$. These SLSF training scenarios are discussed in more detail in Section 3.4, including a formula for the likelihood of fixed scenario $\bar{s}$ occurring, $\mathbb{P}(\bar{s})$. This is used as the weight for SLSF training scenario $\bar{s}$ in the expected objective value (B.1) instead of the equal weighting that we use for LCF training scenarios. The driver responses are recorded as binary $\bar{\boldsymbol{y}}$ variables. The recourse assignment is captured

in the binary $\bar{\boldsymbol{v}}$ variable set. The binary variable set $\bar{\boldsymbol{z}}$ captures the occurrence of *unhappy drivers* as discussed in Section 3.3. Constraints (B.2)-(B.6) and (B.9) are analogous to constraints (3)-(7) and (15) respectively, ensuring a valid menu and recourse assignment. The $\bar{\boldsymbol{z}}$ variable set is not explicitly constrained to be binary, as we explain in more detail in Section 3.3 that this is not necessary. Constraints (B.7) and (B.8) are analogous to the constraints (29) and (30) in our performance analysis formulation (see Section 3.3) respectively, which enforce that each $\bar{z}_{ij}^{\bar{s}}$ has the correct value based on driver behavior $\bar{\boldsymbol{y}}$ and assignment $\bar{\boldsymbol{v}}$.

$$\max_{x,\bar{v},\bar{z}} \quad \frac{1}{\sum_{\bar{s}\in\bar{\Omega}}\mathbb{P}(\bar{s})}\sum_{\bar{s}\in\bar{\Omega}}\mathbb{P}(\bar{s})\sum_{i\in M}\sum_{j\in N}\left((\hat{c}_{ij}-fixcomp_{ij})\bar{v}_{ij}^{\bar{s}}-d_{ij}\bar{z}_{ij}^{\bar{s}}\right) \tag{B.1}$$

$$\text{s.t.} \quad \ell \le \sum_{i\in M} x_{ij} \le \theta \qquad \forall j\in N \tag{B.2}$$

$$\bar{v}_{ij}^{\bar{s}} \le x_{ij} \qquad \forall \bar{s}\in\bar{\Omega},\quad \forall i\in M,\quad \forall j\in N \tag{B.3}$$

$$\bar{v}_{ij}^{\bar{s}} \le \bar{y}_{ij}^{\bar{s}} \qquad \forall \bar{s}\in\bar{\Omega},\quad \forall i\in M,\quad \forall j\in N \tag{B.4}$$

$$\sum_{i\in M}\bar{v}_{ij}^{\bar{s}} \le 1 \qquad \forall \bar{s}\in\bar{\Omega},\quad \forall j\in N \tag{B.5}$$

$$\sum_{j\in N}\bar{v}_{ij}^{\bar{s}} \le 1 \qquad \forall \bar{s}\in\bar{\Omega},\quad \forall i\in M \tag{B.6}$$

$$\bar{z}_{ij}^{\bar{s}} \ge -2+\bar{y}_{ij}^{\bar{s}}+x_{ij}+(1-\sum_{k\in M}\bar{v}_{kj}^{\bar{s}}) \qquad \forall \bar{s}\in\bar{\Omega},\quad \forall i\in M,\quad \forall j\in N \tag{B.7}$$

$$\bar{z}_{ij}^{\bar{s}} \ge 0 \qquad \forall \bar{s}\in\bar{\Omega},\quad \forall i\in M,\quad \forall j\in N \tag{B.8}$$

$$x_{ij},\bar{v}_{ij}^{\bar{s}} \in \{0,1\} \qquad \forall \bar{s}\in\bar{\Omega},\quad \forall i\in M,\quad \forall j\in N \tag{B.9}$$

C Proofs of Lemma 3.1 and Theorem 3.1: Adding Unhappy Driver Variables to LCF

An unhappy driver variable z_{ij}^s, where $z_{ij}^s = 1$ if driver j accepts request i in LCF training scenario s but receives no assignment in scenario s, can be added to LCF, using additional constraints and a penalty term in the objective. We refer to this formulation as LCF-withZ, which is given by (C.1)-(C.3) and (3)-(15). While unhappy drivers are captured in SLSF and the performance analysis, we do not use this formulation instead of LCF as the variable nature of driver selections means unhappy drivers would only occur when the penalties are zero. This is proved in Lemma 3.1. Because of this, the optimal objective value, menus, and compensations for LCF-withZ are the same as for LCF. This is proved in Theorem 3.1.

Lemma 3.1. *Any optimal solution of LCF-withZ can only have positive z_{ij}^{*s} values for $i \in M$ and $j \in N$ such that $d_{ij} = 0$.*

Proof. Suppose $\dot{\boldsymbol{\chi}} = [\ddot{\boldsymbol{p}}^*, \boldsymbol{u}^*, \boldsymbol{v}^*, \boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*, \boldsymbol{\phi}^*]$ is a feasible solution in LCF-withZ. We first show that $\check{\boldsymbol{\chi}} = [\ddot{\boldsymbol{p}}^*, \boldsymbol{u}^*, \boldsymbol{v}^*, \boldsymbol{x}^*, \boldsymbol{y}^* - \boldsymbol{z}^*, \boldsymbol{0}, \boldsymbol{\phi}^*]$, a similar solution that has no unhappy drivers, is also feasible in LCF-withZ with an objective value at least as high as for $\dot{\boldsymbol{\chi}}$. Constraints (3), (6)-(8), and (10)-(14) are met by $\check{\boldsymbol{\chi}}$ as the

$$\max_{\substack{\ddot{p},u,v,\\ x,y,z,\phi}} \quad \frac{1}{|\Omega|}\sum_{s\in\Omega}\sum_{i\in M}\sum_{j\in N}\left(\dot{c}_{ij}v^s_{ij}-\phi^s_{ij}-d_{ij}z^s_{ij}\right) \tag{C.1}$$

s.t. [Constraints (3)-(15)]

$$z^s_{ij} \geq -2+y^s_{ij}+x_{ij}+(1-\sum_{k\in M}v^s_{kj}) \qquad \forall s\in\Omega, \quad \forall i\in M, \quad \forall j\in N \tag{C.2}$$

$$z^s_{ij} \geq 0 \qquad \forall s\in\Omega, \quad \forall i\in M, \quad \forall j\in N \tag{C.3}$$

variable values in these constraints are the same for both $\dot{\boldsymbol{\chi}}$ and $\check{\boldsymbol{\chi}}$. Because $\boldsymbol{z}^*$ is nonnegative, $y^{*s}_{ij}-z^{*s}_{ij} \leq y^{*s}_{ij} \leq x^*_{ij}$, so constraint (4) is upheld by $\check{\boldsymbol{\chi}}$. Similarly, $\frac{1}{|\Omega|}\sum_{s\in\Omega}(y^{*s}_{ij}-z^{*s}_{ij}) \leq \frac{1}{|\Omega|}\sum_{s\in\Omega}y^{*s}_{ij} \leq prem\cdot\ddot{p}^*_{ij}$ so constraint (9) is upheld by $\check{\boldsymbol{\chi}}$. Constraint (5) is met for any $s\in\Omega$, $i\in M$, and $j\in N$ such that $z^{*s}_{ij}=0$, as $y^{*s}_{ij}-z^{*s}_{ij}=y^{*s}_{ij}\geq v^{*s}_{ij}$. For any $s\in\Omega$, $i\in M$, and $j\in N$ such that $z^{*s}_{ij}=1$, we have by the logical nature of constraint (C.2) that $y^{*s}_{ij}=1$, $x^*_{ij}=1$, and $1-\sum_{k\in M}v^{*s}_{kj}=1$, meaning $v^{*s}_{ij}=0$ so $y^{*s}_{ij}-z^{*s}_{ij}=0\geq v^{*s}_{ij}$ so $\check{\boldsymbol{\chi}}$ meets the constraint. As for constraint (15), the x^*_{ij} and v^{*s}_{ij} components are unchanged between solutions and therefore $\check{\boldsymbol{\chi}}$ upholds these components. While not enforced with an explicit constraint, we explain in Section 3.3 that the binary nature of $\boldsymbol{z}^*$ is preserved via the objective value and the unhappy driver constraints ((C.2) and (C.3)). In addition, by the logical nature of (C.2) we have that z^{*s}_{ij} can only equal one if $y^{*s}_{ij}=1$, meaning $z^{*s}_{ij}\leq y^{*s}_{ij}$ for any $s\in\Omega$, $i\in M$, and $j\in N$. These two properties guarantee that $y^{*s}_{ij}-z^{*s}_{ij}$ is binary so constraint (15) is met. We therefore have that $\check{\boldsymbol{\chi}}$ is feasible in LCF-withZ.

The objective value for $\check{\boldsymbol{\chi}}$ is at least as high as the objective value for $\dot{\boldsymbol{\chi}}$, as the $\boldsymbol{v}^*$ and $\boldsymbol{\phi}^*$ terms are the same and $d_{ij}z^{*s}_{ij}\geq 0$ for $\dot{\boldsymbol{\chi}}$ while $d_{ij}\cdot 0=0$ for $\check{\boldsymbol{\chi}}$ for all $s\in\Omega$, $i\in M$, and $j\in N$. This means if $\dot{\boldsymbol{\chi}}$ is an optimal solution, $\check{\boldsymbol{\chi}}$ must also be optimal, so $d_{ij}z^{*s}_{ij}$ must equal zero for all $s\in\Omega$, $i\in M$, and $j\in N$. Therefore, z^{*s}_{ij} can only be positive for $s\in\Omega$, $i\in M$, and $j\in N$ such that $d_{ij}=0$. □

Theorem 3.1. *LCF and LCF-withZ share the same optimal objective value and set(s) of optimal decision variables (**x** and **u**).*

Proof. We first prove that any optimal solution $\dot{\boldsymbol{\chi}}=[\ddot{\boldsymbol{p}}^*,\boldsymbol{u}^*,\boldsymbol{v}^*,\boldsymbol{x}^*,\boldsymbol{y}^*,\boldsymbol{\phi}^*]$ of LCF has an associated solution $\check{\boldsymbol{\chi}}=[\ddot{\boldsymbol{p}}^*,\boldsymbol{u}^*,\boldsymbol{v}^*,\boldsymbol{x}^*,\boldsymbol{v}^*,\boldsymbol{0},\boldsymbol{\phi}^*]$ feasible in LCF-withZ with the same objective value. LCF-withZ constraints (3), (6)-(8), (11), (14), and (10)-(13) are met by $\check{\boldsymbol{\chi}}$ as the variables in these constraints have the same values in $\check{\boldsymbol{\chi}}$ as $\dot{\boldsymbol{\chi}}$. We have $\boldsymbol{v}^*\leq\boldsymbol{y}^*$ from LCF constraint (5); by the transitive property, $\check{\boldsymbol{\chi}}$ upholds LCF-withZ constraints (4) and (9). Because $\boldsymbol{v}^*\leq\boldsymbol{v}^*$, LCF-withZ constraint (5) is upheld. The $\boldsymbol{x}$ and $\boldsymbol{v}$ components of (15) are upheld in LCF-withZ as these are the same for both $\dot{\boldsymbol{\chi}}$ and $\check{\boldsymbol{\chi}}$. The $\boldsymbol{y}$ component is also upheld by $\check{\boldsymbol{\chi}}$ as the optimal $\boldsymbol{y}$ values for $\check{\boldsymbol{\chi}}$ are $\boldsymbol{v}^*$ which are binary (from (15)). As for LCF-withZ constraint (C.2), cancelling out the y^s_{ij} term with v^s_{ij} yields $z^s_{ij}\geq -2+x_{ij}+(1-\sum_{k\in M:k\neq i}v^s_{kj})$. From constraints (6) and (15), we know that $-2+x_{ij}+(1-\sum_{k\in M:k\neq i}v^s_{kj})\leq 0$ so showing $z^{*s}_{ij}\geq 0$ is sufficient to prove the constraint is met. This is the same as constraint (C.3), and as $\boldsymbol{z}^*=\boldsymbol{0}$ for $\check{\boldsymbol{\chi}}$, both (C.2) and (C.3) are upheld so $\check{\boldsymbol{\chi}}$ is feasible in LCF-withZ. As for the objective values of $\dot{\boldsymbol{\chi}}$ in LCF and $\check{\boldsymbol{\chi}}$ in LCF-withZ, the only difference is the $d_{ij}z^s_{ij}$ terms in (C.1). These terms are all zero as $\boldsymbol{z}^*=\boldsymbol{0}$ for $\check{\boldsymbol{\chi}}$ so the objective values are the same.

Next, we show that any optimal solution $\tilde{\boldsymbol{\chi}} = [\ddot{\boldsymbol{p}}^*, \boldsymbol{u}^*, \boldsymbol{v}^*, \boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*, \boldsymbol{\phi}^*]$ of LCF-withZ is also feasible in LCF with the same objective value. The LCF-withZ constraints are a superset of the set of LCF constraints, so $\tilde{\boldsymbol{\chi}}$ is feasible in LCF. Additionally, from Lemma 3.1 we have that z_{ij}^{*s} must be zero or else $d_{ij} = 0$ for all $s \in \Omega$, $i \in M$, and $j \in N$, so all $d_{ij} z_{ij}^s$ terms in (C.1) are zero, leaving the remaining terms in (C.1) identical to the LCF objective formula (2). The objectives therefore have the same value under $\tilde{\boldsymbol{\chi}}$. □

Adding unhappy driver variables to LCF-FM yields LCF-FMwithZ given by (C.1)-(C.3), (4)-(13), and (15) optimized over $\ddot{\boldsymbol{p}}$, $\boldsymbol{u}$, $\boldsymbol{v}$, $\boldsymbol{y}$, $\boldsymbol{z}$ and $\boldsymbol{\phi}$. Using the same logic as the proofs for Lemma 3.1 and Theorem 3.1, like with LCF-withZ, no optimal solution of LCF-FMwithZ has unhappy drivers, and the optimal objective value and first-stage decisions are unchanged.

D Mutated Scenario Generation

A mutated SLSF scenario is created by starting with the scenario with the highest likelihood (found by rounding each p_{ij} value to the nearest integer for each $i \in M$ and $j \in N$) and mutating one driver-request entry at a time (which also affects scenario likelihood (32)) in a random order to match some randomly-generated SLSF training scenario. If the entry in question already matches the random scenario, nothing is changed and the method proceeds to the next entry. This is done until the next mutation would result in the likelihood of the mutated scenario being lower than the likelihood of the most likely scenario by a factor of some predetermined deviation cap. Additional mutated scenarios are generated by mutating the most likely scenario to match entries from new random training scenarios until the desired sample size is achieved. These mutated scenarios all have a likelihood that is approximately the most likely scenario's likelihood divided by the deviation cap, meaning that every scenario will have some significance in the solution.

We use this method to generate SLSF training scenarios instead of equally-weighted random scenarios because early testing showed that both the objective value of the solution and the runtime are better using mutated scenarios. However, the quality of SOL-FMS and SOL-IT solutions does not suffer from using random fixed LCF scenarios, and the fixed LCF scenarios are already equally weighted to match the equally weighted variable LCF scenarios, so LCF fixed scenarios are generated randomly.