Transportation Research Part B 190 (2024) 103082

Contents lists available at ScienceDirect

Transportation Research Part B

journal homepage: www.elsevier.com/locate/trb

Freelance drivers with a decline choice: Dispatch menus in on-demand mobility services for assortment optimization

Yue Yang [a], Seeun William Umboh [b,c], Mohsen Ramezani [a,*]

[a] The University of Sydney, School of Civil Engineering, Sydney, Australia
[b] The University of Melbourne, School of Computing and Information Systems, Melbourne, Australia
[c] ARC Training Centre in Optimisation Technologies, Integrated Methodologies, and Applications (OPTIMA), Australia

ARTICLE INFO

Keywords:
Mobility on-demand
Submodular maximization
Point-to-point transport
Ride-sourcing
Ride-sharing
Crowd-shipping

ABSTRACT

With the prosperity of sharing economy, more part-time and freelance suppliers (i.e., drivers) join on-demand mobility services. Because of suppliers' autonomy and behavioural heterogeneity, the platform cannot ensure that suppliers will accept a dispatch order. One approach to mitigate this supply uncertainty is to provide suppliers with personalized menus of dispatch recommendations. A key issue then is to determine which dispatch orders (that can be passenger or goods services) should be allocated into the assortment menu of each supplier. This paper probabilistically models the suppliers' order acceptance and choice behaviour, including a *decline* option. We propose two assortment optimization problems, disjoint and joint menus, to maximize the expected number of matches. We show that the objective function of the disjoint menu assortment problem is monotone non-decreasing submodular. In contrast, the objective function of the joint menu assortment problem is non-monotone and non-submodular. Accordingly, we present a standard greedy (SG) algorithm to solve the disjoint assortment problem, and γ^*-greedy and local search (LS) algorithms for the joint assortment problem. By bundling orders into consolidated routes, this paper extends the proposed menu assortment methods to the context of meal delivery services. A case study is presented based on the real-world demand in the Manhattan road network. The results show that drivers' autonomy to decline the dispatch orders creates substantial coexistence of idle drivers and unmatched orders in the market. The proposed menu assortment methods curb such matching friction. Moreover, the numerical results demonstrate that the proposed algorithms significantly outperform the traditional dispatching policies applied in practice, e.g., one-to-one matching, in terms of platform efficiency, e.g., achieving more matches, customers' experiences, e.g., reducing waiting time, and benefits for drivers, e.g., tapering off the income inequality among drivers.

1. Introduction

Enabled by the recent technological advances and substantial growth of sharing economy, a plethora of on-demand mobility services, such as ride-hailing (e.g., Uber and Lyft), meal delivery (e.g., Grubhub and Uber Eats), and crowd-sourced logistics (e.g., Deliv and Cargomatic), have reshaped mobility systems worldwide (Dandl et al., 2021; Beojone and Geroliminis, 2021; Alisoltani et al., 2021; Fielbaum et al., 2022). Recent market research valued the global ride-hailing market at USD 113 billion in 2020, anticipating reaching USD 230 billion by 2026. The global online meal delivery market is estimated at USD 22 billion in

* Corresponding author.
E-mail address: mohsen.ramezani@sydney.edu.au (M. Ramezani).

https://doi.org/10.1016/j.trb.2024.103082
Received 25 November 2022; Received in revised form 30 April 2024; Accepted 11 September 2024
Available online 26 September 2024

2020 and is expected to reach USD 49 billion by 2026 (Research and Markets, 2022). These services enable individual customers (e.g., passenger travelling, meal, and grocery delivery, shipping) match and transact with independent transportation suppliers (e.g., drivers, couriers, and mail carriers) (Wang and Yang, 2019; Sampaio et al., 2019; Nourinejad and Ramezani, 2020; Yang and Ramezani, 2022; Jiao and Ramezani, 2022). At the same time, these services allow suppliers to find temporary employment, generate extra income, and increase economic productivity. Dispatching (matching) approaches bridging the customer and supplier sides are crucial to forming the quality of customers' experience and the distribution of suppliers' income (Yan et al., 2020; Xu et al., 2020b). A natural objective of dispatching approaches is to maximize the total number of matches (responses). To this end, the platform would collect available suppliers, request orders periodically, and make dispatch decisions based on various optimization objective functions, constraints, and algorithms.

A prominent feature of transportation network companies (TNC) is leveraging underutilized or excess resources from transportation suppliers, while the companies do not own these resources. It characterizes a sharing economy with freelance drivers sharing their assets (e.g., vehicle, motorcycle, and van) to serve demand requests. In contrast to fixed labour contracts offered in the traditional taxi or freight industry, suppliers in on-demand mobility services are organized through more flexible arrangements. Economic Development Research Group (EDR Group) found that nearly 87% of Uber drivers had another job (income source) when they were working as a ride-hailing driver (Group, 2018). Sui et al. (2019) point out that Didi Chuxing has attracted a variety of regular car owners in addition to licenced taxis in China, and allows them to service request trips (i.e., Didi Express) in their own time. Ramezani et al. (2022) also show that part-time drivers make up 85% of the total supply of Didi Chuxing in Chengdu. Therefore, suppliers in on-demand mobility services are viewed as micro-freelancers and self-employed drivers. Suppliers can make working decisions (i.e., working at their preferred time and area) and have heterogeneous market-behavioural patterns (e.g., being full-time or part-time, being single-homing or multi-homing) based on their earnings and characteristics (Xu et al., 2020a; Ashkrof et al., 2020; de Ruijter et al., 2022; Ramezani et al., 2022; Fielbaum and Tirachini, 2021; Jiao and Ramezani, 2024).

Once suppliers decide to work and select their working time and area, they receive request orders dispatched (or recommended) from platforms. Suppliers might have different preferences (utilities) for different orders. To maximize their utility, they can choose whether to accept or decline their received dispatch orders (Ashkrof et al., 2021). Suppliers' order acceptance behaviour can significantly influence the system performance of on-demand mobility services. As evidence, Xu et al. (2018b) report that over 40% of the ride-hailing requests of Didi Chuxing in Shanghai in 2018 were aborted and received no response from drivers. Hence, it is essential to consider suppliers' order acceptance behaviour and address how to dispatch orders to achieve optimal system performance.

The authority and the ownership of existing on-demand mobility services (e.g., ride-hailing and crowd-sourcing logistics) remain centralized in terms of management, control, and access (Wang and Yang, 2019; Ramezani and Valadkhani, 2023; Valadkhani and Ramezani, 2023). They strive to maximize profit and satisfy the requirement of their customers. To this end, the centralized customer-centric dispatching approaches (e.g., Agatz et al., 2011, 2012; Baldacci et al., 2004; Pelzer et al., 2015; Dickerson et al., 2017; Özkan and Ward, 2020; Lyu et al., 2019; Xu et al., 2018a; Yang et al., 2020b; Qin et al., 2021) are adopted to prioritize the customers' interests (e.g., minimizing the response time and pickup time); however typically without considering the suppliers' willingness to accept the dispatch order. In particular, the platforms apply various information-disclosure policies that share limited orders' information with drivers. In general, order fare and final destination are not shown to drivers before they accept the order. As an example, Uber dispatches orders to drivers without displaying the orders' destination and penalizes the drivers with low acceptance rates (e.g., deprioritizes them in future dispatching rounds) (Cook, 2015). This partial blind policy aims to avoid price-based and destination-based discrimination (Smart et al., 2015) and achieve a high dispatching efficiency. However, it deprives suppliers of autonomy by forcing them to accept the dispatch order, which may lead to dissatisfaction and distrust that can, in turn, influence suppliers' behaviour, especially in accepting orders (Rosenblat and Stark, 2016; Wang et al., 2019, 2021; Ashkrof et al., 2021, 2020).

The aforementioned dispatching policy sparks increasing labour relation tension and worldwide strikes due to the dissatisfaction of drivers with their pay and working conditions from the ride-hailing companies (Isobel Asher Hamilton, 2019). Special attention should be devoted to the suppliers with this mandatory dispatching policy; some freelance suppliers would be reluctant to participate in the market for lack of flexibility and autonomy. They may reject (decline) the order assigned to them. The drivers' autonomy to decline the dispatch orders leads to substantial simultaneous coexistence of idle drivers and unmatched orders in the market, indicating a matching friction between passengers and drivers. Therefore, this paper considers the autonomy and self-interest behaviours of the supply side for designing an effective and efficient dispatching strategy.

A many-to-many dispatching approach can hedge against the restriction of suppliers' autonomy by allowing suppliers to receive multiple orders to choose from Einav et al. (2016) and Fradkin (2017). Mofidi and Pazour (2019), Yang et al. (2019a), Horner et al. (2021), and Ausseil et al. (2022) have demonstrated that judiciously sharing a portion of orders among multiple suppliers is beneficial for maximizing platforms' efficiency and enhancing suppliers' and passengers' experiences. This approach requires providing suppliers with a *menu of orders* to choose from, including a *decline* option. Naturally, increasing the number of orders listed in the menu of individual suppliers increases the chances that they find and select an acceptable order. Notwithstanding, with limited knowledge of suppliers' order acceptance behaviour, this approach can lead to new challenges: (i) some popular (i.e., high utility) orders might receive duplicate selections from drivers, and others receive none, resulting in collisions as only one driver can be finally matched. These collisions tend to deteriorate the platform's efficiency, as the platform could have naturally increased the number of matching by redirecting some of these drivers to other orders. (ii) Drivers perhaps spend a long time for decision-making (i.e., selecting a desirable order) after scrolling through a long list of orders. (iii) Behavioural heterogeneity of the supply side makes

the value of the *decline* option differ from one driver to another, such that the platform cannot predetermine how drivers choose an order from their dispatch menus against their *decline* option.

Motivated by the above discussion, this paper introduces a centralized assortment optimization that offers each driver a tailored dispatch menu in a monopoly on-demand mobility market. The assortment problem is often encountered in retail and supply chain management (for example see Pentico (1974), Hinxman (1980) and Davis et al. (2014)). It requires a delicate balance between providing a broader selection of products with limited inventory for each item and offering a more focused range of options with higher inventory levels for each individual product. In the on-demand mobility market, where suppliers are granted autonomy to select or decline dispatch orders, each customer order can be likened to a product offered to suppliers for their selection. The objective is to determine a specific subset (i.e., dispatch menu) of orders for each supplier to maximize the number of successful matches. Distinct from the general assortment problem, each customer order offered to suppliers in our problem has a stock of *one*, indicating that it is available for matching only once, even if multiple suppliers select the same order. This unique feature introduces inter-driver competition when solving the menu assortment problem: (i) customer orders that receive an excessive number of selections are undesirable, and (ii) suppliers declining the dispatch menu are also undesirable. Consequently, solving this menu assortment problem is challenging because the final matching is determined jointly by the platform's dispatch decisions and the suppliers' interdependent choices. Assume the platform aims to maximize the expected number of matches (i.e., responses). Using a priori knowledge of drivers' order acceptance preferences, we propose two menu assortment problems: disjoint menus, where the drivers are assigned disjoint sets of orders, and joint menus, where the drivers may share orders in their menus. These personalized dispatch menus are recommended to drivers over a given batching window, and orders will be assigned based on the drivers' actual selections.

We model that the utility of suppliers to accept a dispatch order is valued in preferences of order types (e.g., passenger trip, food delivery, and parcel), fare, pickup time, and order's destination. The choice behaviour of suppliers is modelled stochastically and linearly proportional to the utilities of orders in their menus and the utility of the *decline* option. To avoid cross-driver interference and choice collisions, the disjoint menu assortment is proposed to reach the assortment solution without the additional systematic coordination after suppliers' selection. While the disjoint menu assortment is shown to be only sensible for the scenario of the high-value orders (Ashlagi et al., 2022), it becomes increasingly ineffective in the opposite scenarios with a mixture of high- and low-value orders. To achieve better system performance, the joint menu assortment is modelled as a many-to-many assortment problem. The novelty behind the joint model is to maximize the expected number of matches by reducing the collision cost (i.e., situations where the selections of multiple suppliers are duplicated and concentrated on a few high-value orders) and increasing the probability of responding to low-value orders (i.e., listing high-value orders in menus of a few drivers while exposing low-value orders to as many drivers as possible). While the proposed menu assortment methods focus on the single-choice scenario (drivers are only allowed to select at most one order (i.e., item) from their menu), our proposed methods have broader implications for both the shared-mobility and the crowd-sourcing sector. By bundling multiple orders into a single route, our proposed methods optimize the menu assortment problem by modelling each bundle (i.e., route) as a single item in the menu. Our major contributions are as follows:

- We introduce a formulation for solving a broad class of menu assortment problems in on-demand mobility services to maximize the number of realized matches. The problem is grouped into three categories: one-to-one dispatching, disjoint menu assortment, and joint menu assortment.
- We analyse the objective function's submodularity over disjoint and joint menu assortment problems. We show that the objective function of the disjoint model is monotone non-decreasing submodular, while the objective function of the joint model is non-monotone and non-submodular.
- Relying on the property of submodularity proved above, we present two algorithms, standard greedy (SG) and local search (LS), to solve the disjoint and joint menu assortment problems, respectively. The SG algorithm produces an assortment solution at least $1/2$ of the optimal solution in the worst case, while the LS algorithm achieves $1/(3+\varepsilon)$ in the worst case (ε is an arbitrary non-negative number).
- According to the theoretical properties of the homogeneous case of the problem, we also develop a heuristic algorithm named γ^*-greedy for the joint menu assortment problem. The numerical experiments show γ^*-greedy algorithm is capable of reaching the best compromise between computational efficiency and solution quality.
- We propose an application and extension of menu assortment methods specifically tailored for meal delivery operations. By leveraging cluster-based order bundling techniques, our methods dynamically create optimized menus whose items consist of different bundles. Implementing our menu assortment methods result in improved system efficiency, enhanced customer satisfaction, and improved resource utilization.

The layout of this paper is organized as follows. We formally describe the problem and discuss the assumptions in Section 2. Section 3 introduces a general matching model for on-demand mobility services where a menu of orders is offered to drivers, and formulates the disjoint and joint menu assortment models. In Section 4, γ^*-greedy algorithm is first developed based on the analysis of the homogeneous case, and then the disjoint and joint menu assortment problem are solved by standard greedy (SG) and local search (LS) algorithms, respectively. According to the data from Manhattan, New York, Section 5 evaluates the performance of the proposed methods and extends these methods to include order bundling in meal delivery. Concluding remarks are given in Section 6.

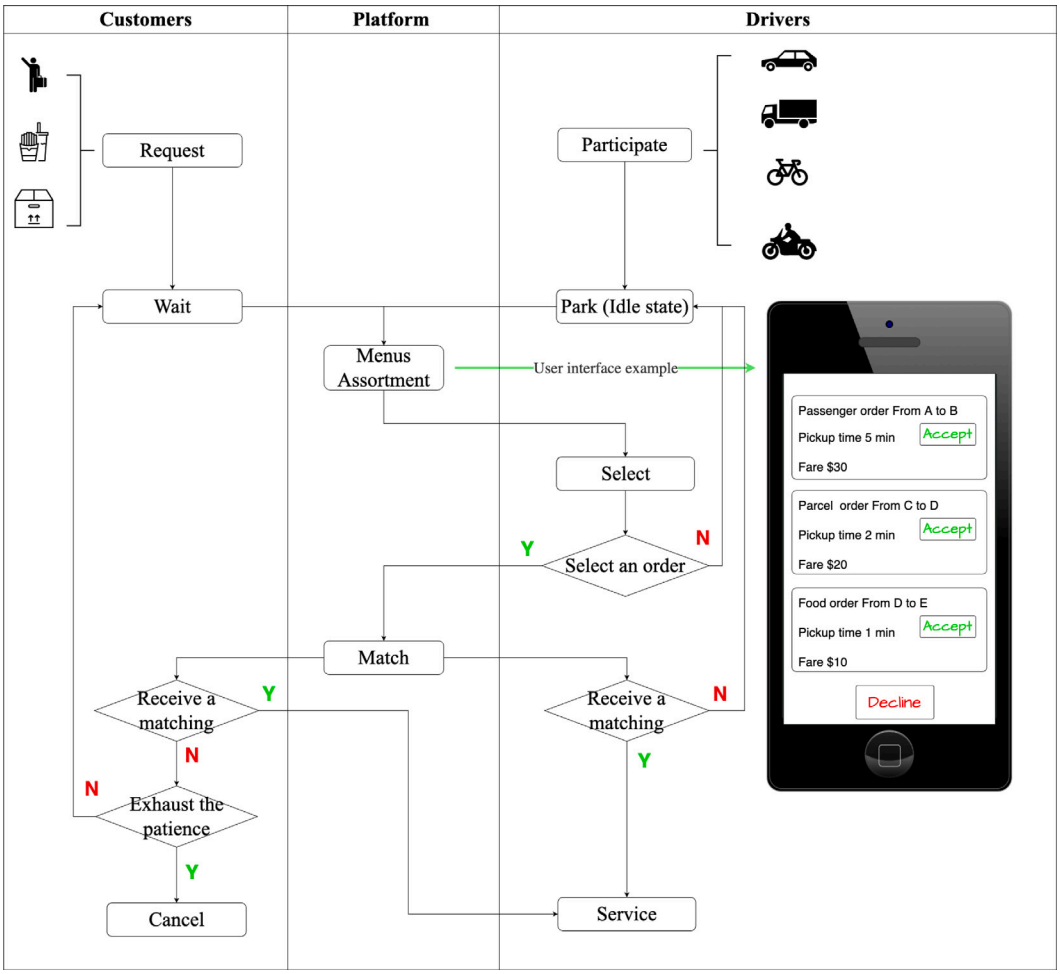


Fig. 1. Schematic framework of the centralized dispatch menu assortment system.

2. Problem description

We consider the interactions among three participants in a monopoly mobility on-demand market, including the platform, drivers, and customers (as illustrated in Fig. 1): (i) **Platform** collects the information of requesting customers (orders) and available idle drivers over a given batching time window. At the end of the window (e.g., every 10 s), the platform employs a centralized approach to offer each driver a personalized dispatch menu. After receiving the response of drivers (who might choose not to accept any orders, the *decline* option), the platform determines the optimal driver–order matching pairs and notifies drivers and customers. (ii) **Drivers** participate in the platform and park (or cruise) when being idle. Once receiving the personalized dispatch menu from the platform, the drivers have the autonomy to select a request order (if any) from the menus or can opt to *decline*. After knowing the matching from the platform, the assigned drivers follow the platform's routing guide to service the corresponding order. At the same time, the unassigned drivers wait in the market for opportunities in the subsequent matching intervals. (iii) **Customers** request an order (e.g., passenger trip or meal delivery) on the platform. One of the following three outcomes occurs after the menus assortment and drivers' selections: a customer's order can be selected by multiple drivers, only one driver, and no drivers. When a customer is selected by at least one driver, this results in a successful match. Otherwise, they will be considered for matching in the following matching intervals until their patience is exhausted and they cancel their request and leave the market.

We assume each order has a set of attributes such as order type (e.g., passenger trip, food delivery, and parcel), fare, and destination location. Each driver–order pair also has a specific pick-up time. These attributes constitute the utility of order o for driver d, $u_{d,o}$, as follows:

$$u_{d,o} = \overbrace{\beta_{0,d}}^{\text{constant}} + \overbrace{\beta_{1,d} \cdot I_o}^{\text{order type}} + \overbrace{\beta_{2,d} \cdot f_o}^{\text{order fare}} - \overbrace{\beta_{3,d} \cdot \tau(l_d, l_o^{\text{org}})}^{\text{pickup time}} + \overbrace{\beta_{4,d} \cdot V(l_o^{\text{dest}})}^{\text{value of the order's destination}} \tag{1}$$

where I_o is the type of order o, f_o represents order o's fare (e.g., priced based on the travel distance and travel time), and $\tau(l_d, l_o^{\text{org}})$ indicates the pick-up time from driver d's current location, l_d, to order o's origin, l_o^{org}. $V(l_o^{\text{dest}})$ is the spatial value function of order o's destination, l_o^{dest}, such that higher $V(\cdot)$ indicates the area is a hotspot to being matched with subsequent orders (Chen et al., 2021; Hamedmoghadam et al., 2019; Ramezani and Nourinejad, 2018). Finally, $\beta_{0,d}$, $\beta_{1,d}$, $\beta_{2,d}$, $\beta_{3,d}$, and $\beta_{4,d}$ are preference parameters of driver d. There are four main factors considered in the utility function: (i) order type: drivers prefer a type of order with higher convenience (e.g., easier to park) and less uncertainty (e.g., less waiting or preparation time at the order's origin). (ii) Order fare: orders with a higher fare will naturally attract the driver to accept them. (iii) Pickup time: a longer pickup time reduces the driver's willingness to choose the order. (iv) Order destination: orders whose destination are in hotspot regions will be more probable to be selected by drivers. Driver d also has a *decline* utility for ignoring all orders in their dispatch menu, denoted by u_d^{c}, which can be estimated using historical behavioural data of driver d.

The problem described in this section can be naturally fitted to shared-mobility (e.g., ride-sharing, ride-pooling, and logistic-sharing) services. Constrained by time windows or detour distances, multiple passengers (meal or parcel orders) can be combined into a bundle serviced by one route. The bundle generation approaches have already been well-studied as variants of the vehicle routing problem (VRP) (Mancini and Gansterer, 2022) and the ride-sharing matching problem (Masoud and Jayakrishnan, 2017b,a; Zhang and Masoud, 2021; Alisoltani et al., 2022; Tafreshian and Masoud, 2020). We can directly employ these techniques to collect all feasible bundles. Then each bundle (i.e., route) can be viewed as an item in the driver menu with a set of attributes (e.g., total fare, pick-up time to the first point of the route, and value of the last drop-off location) to rebuild the utility function. Hence, we have expanded the menu assortment problem to include an order bundling mechanism in meal delivery in Section 5.4.

The defining characteristics and assumptions of the menu assortment problem are: (i) Each driver is assumed to select at most one item out of their menu. (ii) The occupied vehicles will not receive a new dispatch menu before they finish the currently assigned trip. (iii) The platform has prior knowledge of the aggregated driver selection behaviour, which is achieved by estimating average (*expected*) values of $\beta_{0,d}$, $\beta_{1,d}$, $\beta_{2,d}$, $\beta_{3,d}$, $\beta_{4,d}$, and u_d^{c} of all drivers from historical data. (iv) As a tie-breaker rule in case multiple drivers select one order, the order will be assigned to the nearest driver among those who select the order.

3. Problem formulation

At every matching instance, the platform should decide on a menu assortment problem: 'what dispatch menus to be offered to each driver?'. Given a set O of available (unmatched) orders and a set D of idle drivers collected between two successive matching instances, an assortment solution is a subset X of $D \times O$. An element in solution X is a driver–order pair, $(d, o) \in X$ meaning that order $o \in O$ is listed in the menu of driver $d \in D$. Note that pair (d, o) does not mean driver d will be dispatched to service order o. For subset X, we denote $O_d(X) = \{o : (d, o) \in X\}$ to be the menu (i.e., set of orders) that driver $d \in D$ can select from. Similarly, $D_o(X) = \{d : (d, o) \in X\}$ is the set of drivers who have o in their dispatch menus.

Drivers are assumed rational utility maximizers without a commitment to the platform and make decisions independently of other drivers. Therefore, we assume probability $p_{d,o}(X)$ that order o is chosen by driver $d \in D$ from the choice set $O_d(X) \cup \{\text{c}\}$, which is the union of the orders in the dispatch menu and the *decline* option c, is:

$$p_{d,o}(X) = \frac{u_{d,o}}{u_d^{\text{c}} + \sum_{o \in O_d(X)} u_{d,o}}, \forall d \in D, \forall o \in O_d(X), \tag{2}$$

and $p_{d,\text{c}}(X)$ is: the probability that driver $d \in D$ declines the dispatch menu

$$p_{d,\text{c}}(X) = \frac{u_d^{\text{c}}}{u_d^{\text{c}} + \sum_{o \in O_d(X)} u_{d,o}}, \forall d \in D. \tag{3}$$

3.1. General model

For a solution X, we denote that order $o \in O$ is associated with a matching probability $\mathbb{P}_o(X)$ meaning that order o is selected by at least one of the drivers in $D_o(X)$. The expected number of matches $\mathbb{P}(X)$ is the sum of order matching probabilities with X as the solution of the dispatch menu assortment optimization,

$$\mathbb{P}(X) = \sum_{o \in O} \mathbb{P}_o(X). \tag{4}$$

We assume that the choice models of drivers (driver–order and *decline* utilities) are known to the platform. The platform designs a dispatch menu for individual drivers to maximize the expected number of matches $\mathbb{P}(X)$. Formally, we define the menu assortment problem as follows:

$$\max \mathbb{P}(X) \tag{5}$$

$$|D_o(X)| \leq a, \forall o \in O \tag{6}$$

$$|O_d(X)| \leq b, \forall d \in D. \tag{7}$$

The constraint in Eq. (6) describes that each order should be listed in the menus of a drivers at most. The constraint in Eq. (7) limits the maximum size of menus to be less or equal to b (excluding the *decline* alternative) for each driver.

Different values of a and b shape various menu assortment problems: (i) the dispatch menus are disjoint if an order is only offered to at most one driver ($a = 1$). Otherwise, the menus are joint such that an order can be listed in the menus of multiple drivers simultaneously ($a > 1$). (ii) The platform may dispatch one order to each driver ($b = 1$) or display a dispatch menu to each driver ($b > 1$). In summary, we investigate the following three menu assortment problems:

- One-to-one dispatching ($a = 1$, $b = 1$), where each driver is assigned at most one order. This is the common practice studied in the literature.
- Disjoint menu assortment ($a = 1$, $b \geq 1$), where drivers are assigned disjoint dispatch menus, i.e., $O_{d_1}(X) \cap O_{d_2}(X) = \emptyset$, $\forall d_1, d_2 \in D$ and $d_1 \neq d_2$. In particular, one-to-one dispatching is a special disjoint case if $b = 1$.
- Joint menu assortment ($a > 1$, $b > 1$), where multiple drivers may share the same orders in their menus. In particular, any solution X is feasible as long as it satisfies constraints (6) and (7), including one that assigns disjoint menus.

The feasible solution space of the problem in Eq. (5) is exponentially large with the number of orders $|O|$ and drivers $|D|$. Hence, we relax the constraint in Eq. (7) by assuming an infinite menu size ($b = +\infty$).

The above menu assortment problem captures the primary dynamics and trade-offs encountered by a platform when offering drivers dispatch menus to maximize the number of expected matches. The essential step to facilitate successful matching is encouraging drivers to select an order in their dispatch menus. To this end, offering high-value and more orders to drivers increases the chances of them finding an acceptable order over their *decline* option. However, this increases the chances that multiple drivers select similar high-value orders. Therefore, there is a trade-off between encouraging drivers' selection and decreasing duplicate selections (see Section 4.1, which is the homogeneous case).

There are no duplicate selections in the disjoint menu assortment; independent drivers will be matched with their selected order. However, the disjoint menu assortment model is sometimes counterproductive because the number of drivers assigned a non-empty menu (i.e., containing at least one order) can be small especially during a high-supply low-demand time. In contrast, the joint menu assortment increases the number of non-empty menus while it inherently creates a collision when multiple drivers select the same order. Hence, the joint menu assortment model requires a tie-breaker rule to coordinate duplicate selections to achieve the final matching.

3.2. Disjoint menus assortment model

If the menus offered to individual drivers are disjoint ($a = 1$), matching probability $\mathbb{P}_o(X)$ of order o is the selection probability of the driver who have o in their dispatch menu,

$$\mathbb{P}_o(X) = \sum_{d \in D_o(X)} p_{d,o}(X). \tag{8}$$

Without the selection collision in the disjoint menus, individual drivers will get a match as long as they select an order from their dispatch menus. We introduce a selection probability $\mathbb{P}_d(X)$, $\forall d \in D$ that driver d, $\forall d \in D$ does not decline the dispatch menu as

$$\mathbb{P}_d(X) = 1 - p_{d,\mathrm{c}}(X) = \sum_{o \in O_d(X)} p_{d,o}(X) = \frac{\sum_{o \in O_d(X)} u_{d,o}}{u_d^{\mathrm{c}} + \sum_{o \in O_d(X)} u_{d,o}}, \forall d \in D \tag{9}$$

Next, the problem in Eqs. (5)–(7) can be expressed as a many-to-one matching problem:

$$\max \mathbb{P}(X) = \sum_{o \in O} \mathbb{P}_o(X) = \sum_{d \in D} \mathbb{P}_d(X) = \sum_{d \in D} \frac{\sum_{o \in O_d(X)} u_{d,o}}{u_d^{\mathrm{c}} + \sum_{o \in O_d(X)} u_{d,o}} \tag{10}$$

$$|D_o(X)| \leq 1, \forall o \in O. \tag{11}$$

Eq. (10) shows that the objective of the disjoint menus assortment is equivalent to maximizing the sum of selection probabilities over all drivers. Constraint in Eq. (11) guarantees that order o is included in at most one dispatch menu of drivers.

3.3. Joint menus assortment model

Consider the problem in Eqs. (5)–(7) allows orders to be overlapping ($a = \infty$) among multiple dispatch menus. Matching probability $\mathbb{P}_o(X)$ of order o is the probability that order o is selected by at least one driver from $D_o(X)$:

$$\mathbb{P}_o(X) = 1 - \prod_{d \in D_o(X)} \left[1 - p_{d,o}(X)\right] = 1 - \prod_{d \in D_o(X)} \frac{u_d^{\mathrm{c}} + \sum_{o' \in O_d(X)/\{o\}} u_{d,o'}}{u_d^{\mathrm{c}} + \sum_{o' \in O_d(X)} u_{d,o'}}. \tag{12}$$

$\prod_{d \in D_o(X)} \left[1 - p_{d,o}(X)\right]$ represents the probability that none of the drivers in $D_o(X)$ choose order o, where $1 - p_{d,o}(X) = \frac{u_d^{\mathrm{c}} + \sum_{o' \in O_d(X)/\{o\}} u_{d,o'}}{u_d^{\mathrm{c}} + \sum_{o' \in O_d(X)} u_{d,o'}}$. Note that $1 - p_{d,o}(X)$ depends on the other orders, i.e., $o' \in O_d(X)/\{o\}$, listed in the menus of these drivers $d \in D_o(X)$. Eq. (12) suggests increasing the number of orders in $O_d(X)$ would result in a higher value of $1 - p_{d,o}(X)$.

Thus, the joint menus assortment is a many-to-many matching problem without constraint:

$$\max \mathbb{P}(X) = \sum_{o \in O} \mathbb{P}_o(X) = \sum_{o \in O} \left(1 - \prod_{d \in D_o(X)} \frac{u_d^{\mathrm{c}} + \sum_{o' \in O_d(X)/\{o\}} u_{d,o'}}{u_d^{\mathrm{c}} + \sum_{o' \in O_d(X)} u_{d,o'}}\right). \tag{13}$$

4. Analysis and solutions of disjoint and joint menu assortment problems

In this section, we start with an analysis of homogeneous cases (Section 4.1). According to some definitions and mathematical preliminaries (see Appendix A. Mathematical Preliminaries), we propose algorithms with analytical approximation bounds to design disjoint (Section 4.2) and joint (Section 4.3) dispatch menus.

4.1. Homogeneous cases

Consider m orders and n drivers in the system. We assume utilities between driver–order pairs are homogeneous, γu^{c}, where u^{c} is the *decline* utility and the same for all n drivers. Given a menu assortment solution X, $\gamma > 0$ is the ratio between choice probabilities of arbitrary order and the *decline* option:

$$\frac{p_{d,o}(X)}{p_{d,c}(X)} = \frac{u_{d,o}}{u^{c}} = \gamma. \tag{14}$$

We introduce $\mathbb{P}^{\text{Disjoint}}$ and $\mathbb{P}^{\text{Joint}}$ to denote the objective functions of the disjoint and joint models, respectively. Given solution X^{Disjoint} for the disjoint model, we have the objective from Eq. (10)

$$\mathbb{P}^{\text{Disjoint}}(X^{\text{Disjoint}}) = \sum_{d\in D} \mathbb{P}_d = \sum_{d\in D} \frac{|O_d(X^{\text{Disjoint}})|\gamma u^{c}}{u^{c} + |O_d(X^{\text{Disjoint}})|\gamma u^{c}} = \sum_{d\in D} \frac{|O_d(X^{\text{Disjoint}})|\gamma}{1 + |O_d(X^{\text{Disjoint}})|\gamma} \tag{15}$$

where $|O_d(X^{\text{Disjoint}})|$ $(0 \le |O_d(X^{\text{Disjoint}})| \le m/n)$ is the number of orders in the menu of driver d.

Given solution X^{Joint} for the joint menu model, we have the objective from Eq. (13)

$$\begin{aligned}\mathbb{P}^{\text{Joint}}(X^{\text{Joint}}) &= \sum_{o\in O} \mathbb{P}_o = \sum_{o\in O}\left(1 - \prod_{d\in D_o(X^{\text{Joint}})}\left(\frac{u^{c} + (|O_d(X^{\text{Joint}})| - 1)\gamma u^{c}}{u^{c} + |O_d(X^{\text{Joint}})|\gamma u^{c}}\right)\right)\\ &= \sum_{o\in O}\left(1 - \prod_{d\in D_o(X^{\text{Joint}})}\left(\frac{1 + (|O_d(X^{\text{Joint}})| - 1)\gamma}{1 + |O_d(X^{\text{Joint}})|\gamma}\right)\right).\end{aligned} \tag{16}$$

where $|O_d(X^{\text{Joint}})|$ $(0 \le |O_d(X^{\text{Joint}})| \le m)$ is the number of orders in the menu of driver d.

Let the menu of each driver be homogeneous with y $(1 \le y \le m)$ orders $(|O_d| = y, \forall d \in D)$, and let each order be listed in the menu of y $(1 \le y \le n)$ drivers $(|D_o| = y, \forall o \in O)$. Thus, X^{Joint} in the homogeneous case determines the value of y, signifying how many orders should be included in the driver menu or how many drivers can have the order in their menu.

$$\mathbb{P}^{\text{Joint}} = m - m(1 - \frac{\gamma}{1+y\gamma})^{y}, \tag{17}$$

we compute the first-order derivative:

$$\begin{aligned}\frac{\mathrm{d}\mathbb{P}^{\text{Joint}}}{\mathrm{d}y} &= -m\frac{\mathrm{d}(1 - \frac{\gamma}{1+y\gamma})^{y}}{\mathrm{d}y} = -m(1 - \frac{\gamma}{1+y\gamma})^{y}\frac{\mathrm{d}\ln(1 - \frac{\gamma}{1+y\gamma})y}{\mathrm{d}y}\\ &= -m(1 - \frac{\gamma}{1+y\gamma})^{y}\left(\frac{\mathrm{d}y}{\mathrm{d}y}\ln(1 - \frac{\gamma}{1+y\gamma}) + \frac{\mathrm{d}\ln(1 - \frac{\gamma}{1+y\gamma})}{\mathrm{d}y}y\right)\\ &= -m(1 - \frac{\gamma}{1+y\gamma})^{y}\left(\ln(1 - \frac{\gamma}{1+y\gamma}) + y\frac{1}{1 - \frac{\gamma}{1+y\gamma}}\frac{\mathrm{d}(1 - \frac{\gamma}{1+y\gamma})}{\mathrm{d}y}\right)\\ &= -m(1 - \frac{\gamma}{1+y\gamma})^{y}\left(\ln(1 - \frac{\gamma}{1+y\gamma}) + y\frac{1+y\gamma}{1+y\gamma-\gamma}\frac{\gamma^{2}}{(1+y\gamma)^{2}}\right)\\ &= -m(1 - \frac{\gamma}{1+y\gamma})^{y}\left(\ln(1 - \frac{\gamma}{1+y\gamma}) + \frac{\gamma^{2}y}{(1+y\gamma-\gamma)(1+y\gamma)}\right).\end{aligned} \tag{18}$$

Since y must be an integer and fall within the constrained range $\mathbb{Z} \cap [1, \min\{m,n\}]$, finding y^{*} to achieve the maximum value of $\mathbb{P}^{\text{Joint}}$ can be highly challenging. To simplify the problem, we examine two conditions of $\frac{\mathrm{d}\mathbb{P}^{\text{Joint}}}{\mathrm{d}y}$ over the specified domain. First, if $\left(\ln(1 - \frac{\gamma}{1+y\gamma}) + \frac{\gamma^{2}y}{(1+y\gamma-\gamma)(1+y\gamma)}\right) > 0$, $\forall y \in \mathbb{Z} \cap [1, \min\{m,n\}]$, then we have $\frac{\mathrm{d}\mathbb{P}^{\text{Joint}}}{\mathrm{d}y} < 0$, meaning $y = 1$ is the optimal solution for the joint menu problem. Conversely, if $\left(\ln(1 - \frac{\gamma}{1+y\gamma}) + \frac{\gamma^{2}y}{(1+y\gamma-\gamma)(1+y\gamma)}\right) < 0$, we have $\frac{\mathrm{d}\mathbb{P}^{\text{Joint}}}{\mathrm{d}y} > 0$, $\forall y \in \mathbb{Z} \cap [1, \min\{m,n\}]$, which shows that $y = \min\{m,n\}$ is the optimal solution for the joint menu problem.

There are three cases with different numbers of drivers and orders considered.

(1) Balanced case ($m = n$): The optimal solution $X^{\text{Disjoint}^{*}}$ of the disjoint model is straightforward and unique: partitioning m orders into n menus of drivers. That is only one order in the menu of each driver.

$$\mathbb{P}^{\text{Disjoint}}(X^{\text{Disjoint}^{*}}) = \frac{m\gamma}{1+\gamma} = \frac{n\gamma}{1+\gamma}. \tag{19}$$

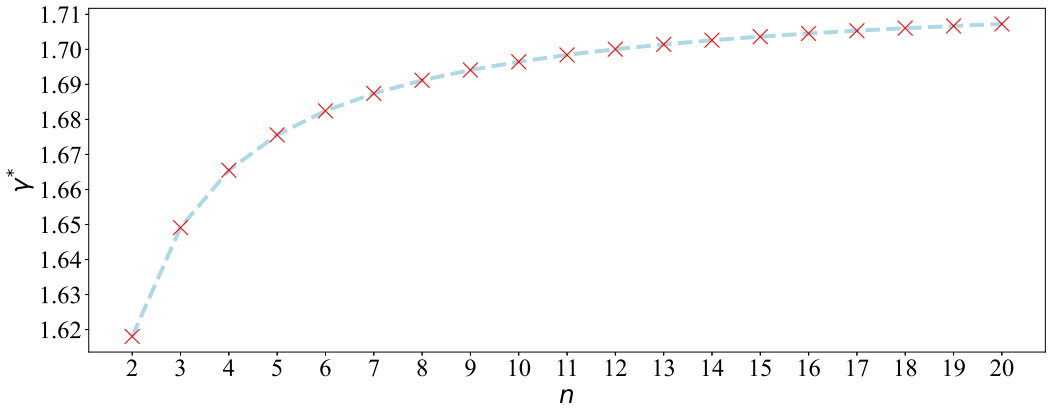


Fig. 2. Given n ranging from 2 to 20, γ^* is solved from Eq. (24). We can see γ^* approaches 1.71 (i.e., $e-1$) as $n \to +\infty$.

According to Eq. (18), the optimal solution of the joint model cannot be solved or analysed directly. We introduce $X^{\text{Joint,Global}}$ that lists all orders in the menu of every driver as the solution of joint model.

$$\begin{aligned}\mathbb{P}^{\text{Joint}}(X^{\text{Joint,Global}}) &= m\left[1-(\frac{u^c+(m-1)\gamma u^c}{u^c+m\gamma u^c})^n\right] = n-n(\frac{u^c+(n-1)\gamma u^c}{u^c+n\gamma u^c})^n \\ &= n-n(\frac{(n-1)\gamma+1}{1+n\gamma})^n = n-n(1-\frac{\gamma}{1+n\gamma})^n\end{aligned} \tag{20}$$

(2) Over-supplied case ($n > m$): The optimal solution X^{Disjoint^*} of the disjoint model is partitioning m orders into m menus of drivers, and

$$\mathbb{P}^{\text{Disjoint}}(X^{\text{Disjoint}^*}) = \frac{m\gamma}{1+\gamma}. \tag{21}$$

The expected number of matches of $X^{\text{Joint,Global}}$ is

$$\mathbb{P}^{\text{Joint}}(X^{\text{Joint,Global}}) = m-m(1-\frac{\gamma}{1+m\gamma})^n \tag{22}$$

(3) Under-supplied case ($n < m$): According to the monotonicity and submodularity proved in Propositions 2 and 3, the optimal solution X^{Disjoint^*} is

$$\mathbb{P}^{\text{Disjoint}}(X^{\text{Disjoint}^*}) = n\cdot\frac{\frac{m}{n}\gamma}{1+\frac{m}{n}\gamma} = \frac{mn\gamma}{n+m\gamma}. \tag{23}$$

The formulation of expected number of matches for $X^{\text{Joint,Global}}$ is consistent with Eq. (22).

By setting the numbers of expected matches of X^{Disjoint^*} and $X^{\text{Joint,Global}}$ equal, we have Eqs. (24), (25), and (26) for the balanced, over-supplied, and under-supplied cases, respectively.

$$n-n(1-\frac{\gamma}{1+n\gamma})^n-\frac{n\gamma}{1+\gamma}=0 \tag{24}$$

$$m-m(1-\frac{\gamma}{1+m\gamma})^n-\frac{m\gamma}{1+\gamma}=0 \tag{25}$$

$$m-m(1-\frac{\gamma}{1+m\gamma})^n-\frac{mn\gamma}{n+m\gamma}=0. \tag{26}$$

There exists a cut-off ratio γ^* delineating the optimality of X^{Disjoint^*} and $X^{\text{Joint,Global}}$. Note that $\gamma > \gamma^*$ indicates the number of expected matches of X^{Disjoint^*} is higher than that of $X^{\text{Joint,Global}}$, while $\gamma \le \gamma^*$ indicates the converse.

Theorem 1. *The cut off ratio γ^* approaches $e-1$ as n limits infinity in the balanced case.*

Proof. We can solve Eq. (24) for any given n and find the cut-off ratio between the optimal objective values of X^{Disjoint^*} and $X^{\text{Joint,Global}}$. Interestingly, when n approaches infinity, the limit of the left part of Eq. (24) is

$$\lim_{n\to+\infty}\left(1-(1-\frac{\gamma}{1+n\gamma})^n-\frac{\gamma}{1+\gamma}\right) = 1-\frac{\gamma}{1+\gamma}-\lim_{n\to+\infty}e^{n\ln(1-\frac{\gamma}{1+n\gamma})}. \tag{27}$$

Let $k = 1/n$, according to L'Hôpital's rule, Eq. (27) is equivalent to

$$\begin{aligned}\lim_{n\to+\infty}\left(1-(1-\frac{\gamma}{1+n\gamma})^n-\frac{\gamma}{1+\gamma}\right) &= 1-\frac{\gamma}{1+\gamma}-\lim_{k\to 0} e^{\frac{1}{k}\ln(\frac{k+\gamma-k\gamma}{k+\gamma})}\\ &= 1-\frac{\gamma}{1+\gamma}-\lim_{k\to 0} e^{\frac{-\gamma^2}{(k+\gamma-k\gamma)(k+\gamma)}}\\ &= 1-\frac{\gamma}{1+\gamma}-e^{-1}\end{aligned} \tag{28}$$

Finally, we could show that $\gamma^* = e-1$ when $n \to +\infty$. This shows that the cut-off ratio γ^* approaches $e-1$ with more orders and more drivers in the system (see Fig. 2). □

According to Eqs. (25) and (26), the cut-off ratio γ^* of the over-supplied and under-supplied cases are presented in Fig. 3 as matrices of blue-red and black-lightorange colour tones, respectively. We observe that γ^* is above $e-1$ in both two cases. Specifically, Fig. 3(a) shows a strong negative correlation between γ^* and m and a positive correlation between γ^* and n, indicating that as m increases or n decreases, γ^* tends to decrease. An intriguing observation also arises in the top-right of Fig. 3(a): in over-supplied scenarios with a sharper supply–demand ratio, γ^* becomes exceedingly large (e.g., surpassing 1 000 000 for $m = 2$, $n = 20$). In contrast, Fig. 3(b) suggests that an increasing trend and a decreasing trend of γ^* with respect to m and n exist in the under-supplied case.

The analysis of above three cases indicates X^{Disjoint^*} is effective if the demand consists of high-value ($\gamma > \gamma^*$) orders, otherwise, $X^{\text{Joint,Global}}$ can achieve a higher performance. In addition, γ^* changes with the number of orders and drivers in the system. Inspired by the discussion of γ^*, we introduce γ^*-greedy algorithm in Algorithm 1 for the joint menu model considering heterogeneous drivers and orders. We introduce utility ratio $\gamma_{d,o}$ for $\forall d \in D$, $\forall o \in O$

$$\gamma_{d,o} = \frac{u_{d,o}}{u_{d,\mathrm{c}}}. \tag{29}$$

Built on the foundation of Theorem 1, Algorithm 1 aims to solve the joint menu assortment by iteratively finding the maximum $\gamma_{d,o}$ among the driver–order pairs. The algorithm 1 comprises of the following four steps:

- *Initialization (line 1)*: Algorithm 1 sets solution set X as $\emptyset$, generates two set O' and D' to duplicate O and D, and finds driver–order pair (d^*, o^*) with the maximum γ.
- *Disjoint dispatching (lines 2-7)*: If $\gamma_{d^*,o^*} > \gamma^*$, Algorithm 1 updates X by including x_{d^*,o^*}, then order o^* and driver d^* are removed from O' and D', respectively, and driver–order pair (d^*, o^*) will be found from remaining orders O' and drivers D'. Since o^* is uniquely matched with d^* in X, and vice versa, we call this process "*Disjoint dispatching*". This process will be continued until $\gamma_{d^*,o^*} \le \gamma^*$.
- *Global dispatching (lines 8-12)*: Algorithm 1 continues to update X by listing every order in remaining set O' into the menu of every driver in D.
- *Output*: Algorithm 1 outputs the final solution X.

Recognizing that the computation time of the joint menu assortment may scale exponentially with a high volume of request orders and idle drivers, the purpose of the γ^*-greedy algorithm is to strike a balance between solution quality and computational efficiency. The time complexity of this algorithm is $\mathbf{O}(|O|^2 \times |D|)$. Note that although the γ^*-greedy algorithm originates from mathematical findings ($\gamma^* = e-1$) of the homogeneous case, it reveals a fundamental pattern existing in both homogeneous and heterogeneous cases: the disjoint model outperforms in scenarios abundant with high-value orders (if $\gamma_{d,o} > \gamma^*$), while the joint model performs better in scenarios dominated by low-value orders (if $\gamma_{d,o} < \gamma^*$). However, due to the increased complexity of the heterogeneous case with varying utilities and menu sizes, finding the γ^* becomes challenging and impractical. Hence, we adopt $\gamma^* = e-1$ to maintain simplicity in the algorithm. As a supplementary material in Appendix E. Adaptive γ^*-greedy algorithm, we also propose an adaptive algorithm by computing γ^* according to different numbers of drivers and orders. The numerical results further offer compelling evidence supporting the choice of $\gamma = e-1$ in the γ^*-greedy algorithm.

4.2. Disjoint menus algorithm

In this section, we show that selection probability $\mathbb{P}_d(X)$, $\forall d \in D$ satisfies the following propositions:

Proposition 1. *$\mathbb{P}_d(X)$, $\forall d \in D$ are a normalized function that $\mathbb{P}_d(\emptyset) = 0$.*

Proposition 2. *$\mathbb{P}_d(X)$, $\forall d \in D$ are monotone non-decreasing.*

Proof. Given two arbitrary subsets S and T, and $S \subseteq T \subseteq \mathbb{X}$, we have

$$\mathbb{P}_d(S) = \frac{\sum_{o\in O_d(S)} u_{d,o}}{u_d^{\mathrm{c}} + \sum_{o\in O_d(S)} u_{d,o}}, \mathbb{P}_d(T) = \frac{\sum_{o\in O_d(T)} u_{d,o}}{u_d^{\mathrm{c}} + \sum_{o\in O_d(T)} u_{d,o}} \tag{30}$$

We know that $O_d(S) \subseteq O_d(T) \subseteq O$ for arbitrary driver d. Thus, we have $\sum_{o\in O_d(S)} u_{d,o} \le \sum_{o\in O_d(T)} u_{d,o}$ for $d \in D$. According to the fact that $\frac{a}{b} \le \frac{a+c}{b+c}$ for any real positive numbers $a < b$, and c, we can complete the proof of $\mathbb{P}_d(S) \le \mathbb{P}_d(T)$. □

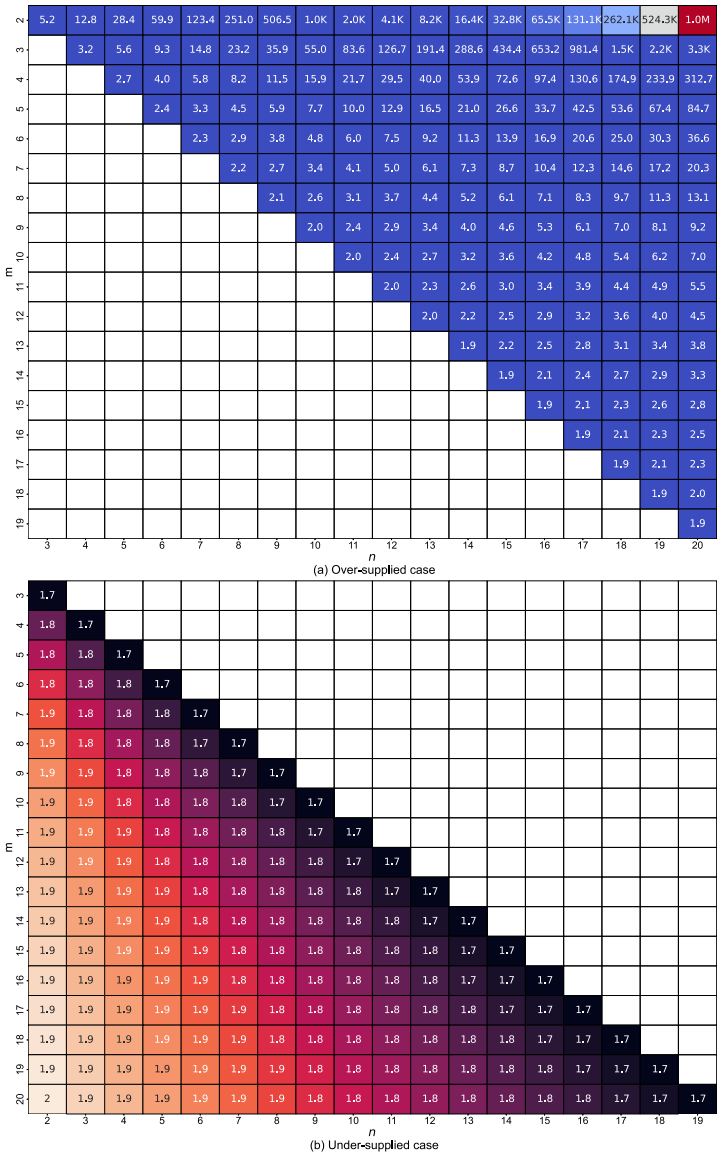


Fig. 3. Heatmap chart of γ^* with different m orders and n drivers. (a) the over-supplied case and (b) the under-supplied case.

Algorithm 1 γ^*-greedy algorithm

Input: O, D
Output: X

$X = \emptyset$, $O' = O$, $D' = D$, $\gamma^* = e - 1$, $(d^*, o^*) = \text{argmax}_{(d,o)}\{\gamma_{d,o}\}$
while $\gamma_{d^*,o^*} > \gamma^*$ **do**
    $X \leftarrow X \cup \{x_{d^*,o^*}\}$
    $O' \leftarrow O' \backslash \{o^*\}$
    $D' \leftarrow D' \backslash \{d^*\}$
    $(d^*, o^*) = \text{argmax}_{(d,o)}\{\gamma_{d,o}\}$
end while
for $o \in O'$ **do**
    for $d \in D$ **do**
        $X \leftarrow X \cup \{x_{d,o}\}$
    end for
end for

Proposition 3. *$\mathbb{P}_d(X)$, $\forall d \in D$ are submodular.*

Proof. First we consider function $F(y) = \frac{y}{c+y}$ where y is a non-negative continuous variable and c is a non-negative constant independent of y. We have:

$$\frac{\mathrm{d}F(y)}{\mathrm{d}y} = \frac{c}{(c+y)^2} > 0 \tag{31}$$

$$\left(\frac{\mathrm{d}^2F(y)}{\mathrm{d}y^2}\right) = \frac{-2c(c+y)}{(c+y)^4} < 0 \tag{32}$$

Thus we show $F(y)$ is a concave function. Given two arbitrary subsets S and T that S is contained in T, $S, T \subseteq \mathbb{X}$, and a feasible driver–order pair (i,j), $(i,j) \notin S$, $(i,j) \notin T$, we have

- If i is not d,

$$\mathbb{P}_d(T \cup \{(i,j)\}) - \mathbb{P}_d(T) = 0 \tag{33}$$

$$\mathbb{P}_d(S \cup \{(i,j)\}) - \mathbb{P}_d(S) = 0 \tag{34}$$

- If i is d,

$$\begin{aligned}\mathbb{P}_d(T \cup \{(i,j)\}) - \mathbb{P}_d(T) &= \mathbb{P}_d(T \cup \{(d,j)\}) - \mathbb{P}_d(T) \\ &= \frac{\sum_{o\in O_d(T)} u_{d,o} + u_{d,j}}{u_d^{\mathrm{c}} + \sum_{o\in O_d(T)} u_{d,o} + u_{d,j}} - \frac{\sum_{o\in O_d(T)} u_{d,o}}{u_d^{\mathrm{c}} + \sum_{o\in O_d(T)} u_{d,o}}\end{aligned} \tag{35}$$

$$\begin{aligned}\mathbb{P}_d(S \cup \{(i,j)\}) - \mathbb{P}_d(S) &= \mathbb{P}_d(S \cup \{(d,j)\}) - \mathbb{P}_d(S) \\ &= \frac{\sum_{o\in O_d(S)} u_{d,o} + u_{d,j}}{u_d^{\mathrm{c}} + \sum_{o\in O_d(S)} u_{d,o} + u_{d,j}} - \frac{\sum_{o\in O_d(S)} u_{d,o}}{u_d^{\mathrm{c}} + \sum_{o\in O_d(S)} u_{d,o}}\end{aligned} \tag{36}$$

According to concavity of $F(y)$, we shall have:

$$\mathbb{P}_d(T \cup \{(i,j)\}) - \mathbb{P}_d(T) \le \mathbb{P}_d(S \cup \{(i,j)\}) - \mathbb{P}_d(S) \tag{37}$$

Therefore, we show $\mathbb{P}_d(X)$, $\forall d \in D$ are submodular. □

Proposition 1 means that when orders are not shown to drivers d, $\forall d \in D$, matching probabilities $\mathbb{P}_d(X)$, $\forall d \in D$ are zero. Proposition 2 means that matching probabilities $\mathbb{P}_d(X)$, $\forall d \in D$ are nondecreasing as more orders are included in the menu of drivers. As a result of Propositions 1 and 2, matching probabilities $\mathbb{P}_d(X)$, $\forall d \in D$ will always be nonnegative. Proposition 3 is the submodularity, which indicates that less marginal probability can be brought by adding a new order in the dispatch menu of d, $\forall d \in D$.

Our analysis shows that the disjoint menus assortment problem is a submodular maximization problem under a partition matroid constraint, which is a variant of Problem (1.2) in Fisher et al. (1978). Maximizing a submodular function under a matroid constraint is a member of the class of NP-hard problems (Vondrák, 2007). In particular, approximation ratios with lower bounds, compared to the optimal solution, are generally used to measure the performance of the algorithms. Under a uniform matroid constraint, some papers presented an improved approximation ratio of $(1 - 1/e)$ using the greedy-based algorithms (Fisher et al., 1978; Nemhauser et al., 1978). For the submodular maximization problem under a partition matroid constraint, the standard greedy algorithm (SG) (Conforti and Cornuéjols, 1984) is widely employed to solve the problem with an approximation ratio of $1/2$ and is detailed in Algorithm 2:

- *Initialization (line 1)*: Algorithm 2 initializes solution set $X^0 = \emptyset$ and feasible solution space $V^0 = \mathbb{X}$.
- *Addition (lines 2–7)*: Algorithm 2 updates solution set X^{k+1} by adding element $x \in V^k$ with the maximum marginal objective improvement $\mathbb{P}(X^k \cup \{x\}) - \mathbb{P}(X^k)$ sequentially. Then Algorithm 2 constructs V^{k+1} consisting of driver–order pairs satisfying the constraints in Eq. (11).
- *Output (line 8)*: Algorithm 2 outputs the final solution X^k.

The time complexity of this algorithm is $\mathbf{O}(|O|^2 \times |D|)$. The algorithm has the following performance guarantee, proved in Conforti and Cornuéjols (1984):

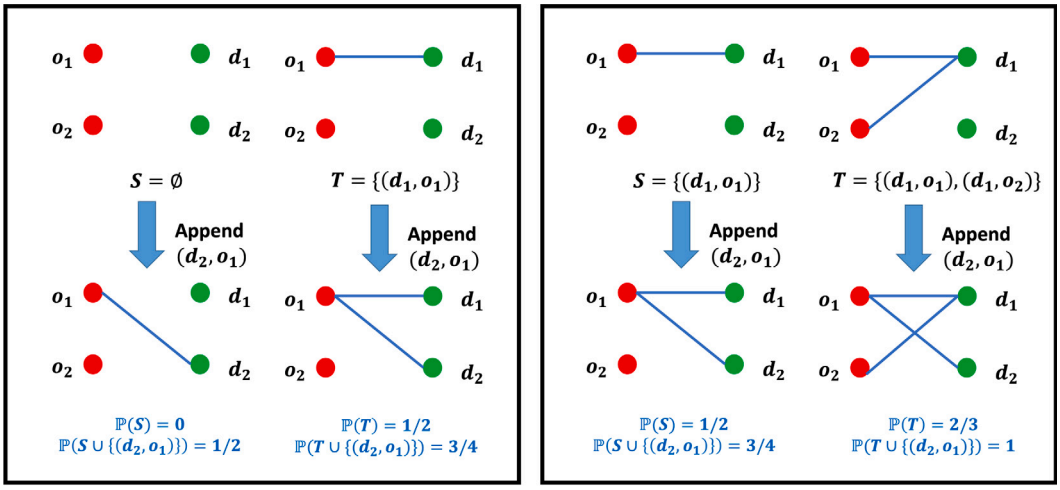
Theorem 2. *Assume Propositions 1, 2, and 3 hold. Let X^* be an optimal solution of the problem in Eq. (10), and X^{SG} be the solution produced by SG algorithm. Then the approximation ratio can be derived from the total curvature $c_{\mathbb{P}}$*

$$r(X^{SG}) := \frac{\mathbb{P}(X^{SG})}{\mathbb{P}(X^*)} \ge \frac{1}{c_{\mathbb{P}} + 1} \tag{38}$$

In the worst case ($c_{\mathbb{P}} = 1$), the algorithm achieves objective function at least 1/2 of the optimal value.

Algorithm 2 Standard Greedy (SG) Algorithm

Input: O, D
Output: X^k
1: $k = 0$, $X^0 = \emptyset$, $V^0 = \mathbb{X}$
2: **while** $V^k \neq \emptyset$ **do**
3: $x_k = \text{argmax}_{x \in V^k}(\mathbb{P}(X^k \cup \{x\}) - \mathbb{P}(X^k))$
4: $X^{k+1} \leftarrow X^k \cup \{x_k\}$
5: $V_{k+1} \leftarrow \{x \in \mathbb{X}/X_{k+1} : \{x\}$ satisfies Eq. (11)$\}$
6: $k \leftarrow k + 1$
7: **end while**
8: $X^{SG} \leftarrow X^k$



(a) $\mathbb{P}(S \cup \{(d_2, o_1)\}) - \mathbb{P}(S) \geq \mathbb{P}(T \cup \{(d_2, o_1)\}) - \mathbb{P}(T)$ (b) $\mathbb{P}(S \cup \{(d_2, o_1)\}) - \mathbb{P}(S) \leq \mathbb{P}(T \cup \{(d_2, o_1)\}) - \mathbb{P}(T)$

Fig. 4. Suppose there are two drivers and two orders, utilities between arbitrary driver–order pair and the *decline* utilities of two drivers are assumed to be homogeneous and equal to 1. We introduce two examples of appending (d_2, o_1) into sets S and T in (a) and (b). Note that $S \subseteq T$. In (a), $S = \emptyset$ and $T = \{(d_1, o_1)\}$, $\mathbb{P}(T \cup \{(i,j)\}) - \mathbb{P}(T) = 0.25 < \mathbb{P}(S \cup \{(i,j)\}) - \mathbb{P}(S) = 0.5$. In (b), $S = \{(d_1, o_1)\}$ and $T = \{(d_1, o_1), (d_1, o_2)\}$, $\mathbb{P}(T \cup \{(i,j)\}) - \mathbb{P}(T) = 0.33 > \mathbb{P}(S \cup \{(i,j)\}) - \mathbb{P}(S) = 0.25$.

4.3. Joint menus algorithm

In this section, we show that objective function $\mathbb{P}(X)$ in Eq. (13) satisfies the following propositions:

Proposition 4. *$\mathbb{P}(X)$ is non-monotone.*

Proof. Given arbitrary feasible sets S and $S \cup \{(i,j)\}$, $S \subseteq \mathbb{X}$, and $S \cup \{(i,j)\} \subseteq \mathbb{X}$, we have

$$\mathbb{P}(S \cup \{(i,j)\}) - \mathbb{P}(S) = \mathbb{P}_j(S \cup \{(i,j)\}) - \mathbb{P}_j(S) + \sum_{o \in O_i(S)} \left(\mathbb{P}_o(S \cup \{(i,j)\}) - \mathbb{P}_o(S)\right) \tag{39}$$

where $\mathbb{P}_j(S \cup \{(i,j)\}) - \mathbb{P}_j(S)$ is the change of $\mathbb{P}_j$ after inserting (i,j) into S, and $(\mathbb{P}_o(S \cup \{(i,j)\}) - \mathbb{P}_o(S))$ is the change of matching probabilities of orders (excluding j) existing in driver i's current menu $O_i(S)$,

$$\mathbb{P}_j(S \cup \{(i,j)\}) - \mathbb{P}_j(S) = \frac{u_{i,j}}{u_i^c + \sum_{o \in O_i(S)} u_{i,o} + u_{i,j}} \times \prod_{d \in D_j(S)} \left[1 - p_{d,j}(S)\right] \geq 0, \tag{40}$$

$$\mathbb{P}_o(S \cup \{(i,j)\}) - \mathbb{P}_o(S) = \left(\frac{u_{i,o}}{u_i^c + \sum_{o' \in O_i(S)} u_{i,o'} + u_{i,j}} - \frac{u_{i,o}}{u_i^c + \sum_{o' \in O_i(S)} u_{i,o'}} \right) \times \prod_{d' \in D_o(S)} \left[1 - p_{d',o}(S)\right] \leq 0 \tag{41}$$

Eq. (40) indicates that matching probability of order j increases with the number of drivers whose dispatch menu includes j. However, Eq. (41) points out that matching probabilities of order o, $\forall o \in O_i(S)$ decreases if we introduce j into i's dispatch menu $O_i(S)$. When we revisit Eq. (39), we could show that $\mathbb{P}(S \cup \{(i,j)\}) - \mathbb{P}(S)$ is not always negative or positive, which depends on the specific values of $u_{i,j}$, $u_{i,o}, \forall o \in O_i(S)$, and S. Hence, $\mathbb{P}(X)$ is non-monotone. □

Proposition 5. *$\mathbb{P}(X)$ is neither submodular nor supermodular.*

Proof. Fig. 4 illustrates two examples of appending (d_2, o_1) into S and T, we can see that $\mathbb{P}(T \cup \{(i,j)\}) - \mathbb{P}(T) \leq \mathbb{P}(S \cup \{(i,j)\}) - \mathbb{P}(S)$ or $\mathbb{P}(T \cup \{(i,j)\}) - \mathbb{P}(T) \geq \mathbb{P}(S \cup \{(i,j)\}) - \mathbb{P}(S)$ cannot simultaneously hold in two examples. Hence, $\mathbb{P}(X)$ is neither submodular nor supermodular. □

Proposition 4 means that the expected number of matches might fall down if we include more orders into the dispatch menus of drivers. Proposition 5 means that the objective of the joint menus assortment does not possess submodularity nor supermodularity. To solve the non-monotone non-submodular maximization problem in Eq. (13), we present a local search algorithm (Feige et al., 2011) in Algorithm 3. We begin with the definition of approximate local optimum λ.

Definition 1 (*Approximate Local Optimum*)**.** Given a set function $f : 2^{\mathbb{X}} \to R$, set $S \subseteq \mathbb{X}$ is called $(1+\lambda)$-approximate local optimum if

$$f(S \cap \{s\}) \leq (1+\lambda) \times f(S), \forall s \in S \tag{42}$$

$$f(S \cup \{s\}) \leq (1+\lambda) \times f(S), \forall s \notin S \tag{43}$$

Algorithm 3 executes four steps:

- *Initialization (line 1)*: Algorithm 3 initializes solution set X^0 with finding the driver–order pair to maximize the objective function.
- *Addition (lines 2–5)*: Algorithm 3 constructs the solution set X^{k+1} of iteration $k+1$ by selecting element $x \in \mathbb{X} \backslash X^k$ to add into X^{k+1} if x satisfies $\mathbb{P}(X^k \cup \{x\}) \geq (1+\lambda)\mathbb{P}(X^k)$.
- *Deduction (lines 6–9)*: Algorithm 3 updates solution set X^{k+1} of iteration $k+1$ by removing $x \in X^k$ satisfying $\mathbb{P}(X^k \cap \{x\}) \geq (1+\lambda)\mathbb{P}(X^k)$.
- *Output (line 10)*: Algorithm 3 selects the final solution between X^k and $\mathbb{X} \backslash X^k$ achieving the higher objective.

Algorithm 3 Local Search (LS) Algorithm

Input: O, D, λ
Output: X^k
1: $k = 0$, $X^0 = \{x : x = \text{argmax}_x \mathbb{P}(\{x\}), \forall x \in \mathbb{X}\}$
2: **while** there exists $x \in \mathbb{X} \backslash X^k$ such that $\mathbb{P}(X^k \cup \{x\}) \geq (1+\lambda)\mathbb{P}(X^k)$ **do**
3: $X^{k+1} \leftarrow X^k \cup \{x\}$
4: $k \leftarrow k+1$
5: **end while**
6: **if** there exists $x \in X^k$ such that $\mathbb{P}(X^k \backslash \{x\}) \geq (1+\lambda)\mathbb{P}(X^k)$ **then**
7: $X^k \leftarrow X^k \backslash \{x\}$
8: Go back to the **while** loop in line 2
9: **end if**
10: $X^{\text{LS}} \leftarrow \text{argmax}\{\mathbb{P}(X^k), \mathbb{P}(\mathbb{X} \backslash X^k)\}$

The time complexity of Algorithm 3 is $\mathbf{O}(|O|^2 \times |D|^2)$. The performance guarantee of Algorithm 3 is given by:

Theorem 3. *Given a positive number $\varepsilon > 0$, $\check{c}_{\mathbb{P}} \in [0,1)$, and $\alpha \in [0,1]$, let X^{LS} be the returned solution of Algorithm 3 and let X^* be any optimum solution of the problem in Eq. (13). The approximation ratio can be proved:*

$$r(X^{LS}) := \frac{\mathbb{P}(X^{LS})}{\mathbb{P}(X^*)} \geq \frac{\alpha^2}{3+\varepsilon\alpha^2}. \tag{44}$$

In the worst case ($\alpha = 1$), the algorithm achieves at least $1/(3+\varepsilon)$ of the optimal objective value.

Proof. Consider an optimal solution X^*, and $\lambda \in (0,1)$. If Algorithm 3 terminates, the solution X^{LS} obtained at the end is a $(1+\lambda)$-approximate local optimum. By Definition 1, we have

$$\begin{aligned} &\mathbb{P}(X^{\text{LS}} \backslash \{s\}) \leq (1+\lambda)\mathbb{P}(X^{\text{LS}}) \\ &\mathbb{P}(X^{\text{LS}} \cup \{s\}) \leq (1+\lambda)\mathbb{P}(X^{\text{LS}}). \end{aligned} \tag{45}$$

Using Definition 6 and arbitrary driver–order pair s, $s \in X^{\text{LS}} \backslash X^*$, we have

$$(1-\check{c}_{\mathbb{P}}) \cdot (\mathbb{P}(X^{\text{LS}} \cup X^*) - \mathbb{P}(X^{\text{LS}} \cup X^* \backslash \{s\})) \leq \mathbb{P}(X^{\text{LS}}) - \mathbb{P}(X^{\text{LS}} \backslash \{s\}). \tag{46}$$

Gathering these inequalities, we get

$$\begin{aligned} 2(1+\lambda)\mathbb{P}(X^{\text{LS}}) + \mathbb{P}(X^{\text{LS}}) - \mathbb{P}(X^{\text{LS}} \backslash \{s\}) &\geq \mathbb{P}(X^{\text{LS}} \backslash \{s\}) + \mathbb{P}(X^{\text{LS}} \cup \{s\}) + (1-\check{c}_{\mathbb{P}}) \cdot (\mathbb{P}(X^{\text{LS}} \cup X^*) - \mathbb{P}(X^{\text{LS}} \cup X^* \backslash \{s\})) \\ &\geq \mathbb{P}(\{s\}) + \mathbb{P}(X^{\text{LS}}) + (1-\check{c}_{\mathbb{P}}) \cdot (\mathbb{P}(X^{\text{LS}} \cup X^*) - \mathbb{P}(X^{\text{LS}} \cup X^* \backslash \{s\})) \end{aligned} \tag{47}$$

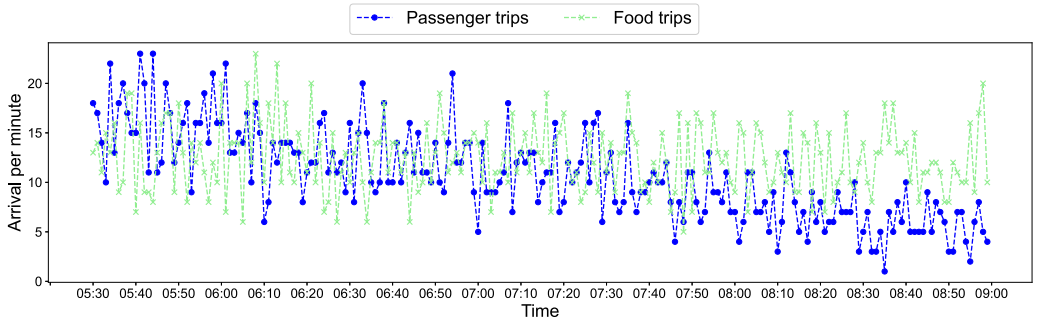


Fig. 5. Average number of demand arrivals per minute of the first two weeks in December 2020.

Thus, we have,

$$2(1+\lambda)\mathbb{P}(X^{\mathrm{LS}}) \geq \mathbb{P}(\{s\}) + \mathbb{P}(X^{\mathrm{LS}} \backslash \{s\}) + (1-\check{c}_{\mathbb{P}}) \cdot (\mathbb{P}(X^{\mathrm{LS}} \cup X^{*}) - \mathbb{P}(X^{\mathrm{LS}} \cup X^{*} \backslash \{s\})) \tag{48}$$

Given $\lambda = \frac{\alpha^2 \varepsilon}{(1-\check{c}_{\mathbb{P}}) \cdot |O| \cdot |D|}$ for Inequality (48), Yang et al. (2019b) proved that $\mathbb{P}(X^{\mathrm{LS}}) \geq \frac{\alpha^2}{3+\varepsilon\alpha^2} \cdot \mathbb{P}(X^{*})$. □

5. Numerical experiments

In this section, we examine the performance of the proposed algorithms and demonstrate the benefits of dispatch menu assortment. All the experiments are conducted in a simulation environment in which Manhattan island is considered. The road network of Manhattan is extracted from OpenStreetMap, which comprises 6533 nodes and 10,206 directed links (road segments), including streets, highways, bridges, and tunnels. The adjacency matrix, the shortest paths, and the shortest travel times are pre-calculated and stored in look-up tables. It is assumed that vehicles travel at the same speed of 8.5 [m/s] in each road segment. The experiments are conducted for four hours (evening peak) between 05:00 PM and 09:00 PM, including the first 30 min (05:00 PM–05:30 PM) as a warm-up period for the network to be populated with demand and supply.

5.1. Demand and supply

The demand data consist of passenger trips and food trips. Passenger trips are obtained from taxi datasets in December 2020 from Yellow Cab's website.[1] The key data fields include the origin and destination geocoordinates (latitude and longitude), order fare, and order request time stamp. Trips with abnormal geocoordinates, e.g., outside the spatial boundary, and with unrealistic distance, e.g., shorter than 0.32 [km] or longer than 15 [km], are eliminated from the data. Then origins and destinations are projected to the closest nodes of the road network. Each passenger is assigned a matching patience time stochastically drawn from a truncated Gaussian distribution in the range of 0.5 [min] to 1 [min] with a mean of 0.75 [min] and a standard deviation of 0.25 [min]. In addition, Appendix B. Synthesizing food orders details the process of generating food trips from 477 real-world restaurants in Manhattan. Each food requester is also assigned a matching patience time stochastically drawn from a truncated Gaussian distribution in the range of 5 [min] to 10 [min] with a mean of 7.5 [min] and a standard deviation of 2.5 [min]. Fig. 5 shows the number of arriving orders on average per minute for the first two weeks in December 2020. The graph illustrates a considerable time-varying pattern of the arriving trip demand.

The values of I_o (see Eq. (1)) are transformed from nominal codes (i.e., passenger or food) into numerical codes (i.e., 1 or 0) before they are used for utility estimation. This transformation is because a food trip often entails higher uncertainty considering the preparation time and parking time (Sungur et al., 2010). In addition, we show the histograms of fare per passenger trip and food trip in Fig. 6. We can observe that over 90% of fares of the trip and food orders are less than 16.0 and 7.4 [USD], respectively.

Fig. 7 shows the origin–destination heat-maps of passenger and food orders. The passenger pickup areas are spread in Manhattan, while the food delivery pickups are mainly around the midtown area at Penn Station and Times Square, and on the South West Side. This is expected since the geographical locations of selected restaurants are mainly distributed in the centre and the south. Since thousands of nodes are considered in the road network, we adopt a space discretization method to partition the Manhattan network into zones with 672 similar hexagons with an approximately 340 [m] diagonal length to evaluate the value of trip destinations. The spatial values of order destinations $V(I_o^{\mathrm{dest}})$ are measured by the normalized count of (historical) pickups in each hexagon. Note that the value of $V(I_o^{\mathrm{dest}})$ is uniformly scaled between 0 and 1. An order whose destination is in a more dense pickup hexagon has a higher $V(I_o^{\mathrm{dest}})$.

Two hundred drivers are initially generated in the road network at 05:00 PM randomly. The number and location of new arriving drivers are assumed to be stochastic and time-varying to replicate an erratic realistic feature in the first hour (05:00 PM–06:00 PM).

[1] https://www1.nyc.gov/site/tlc/about/tlc-request-record-data.page.

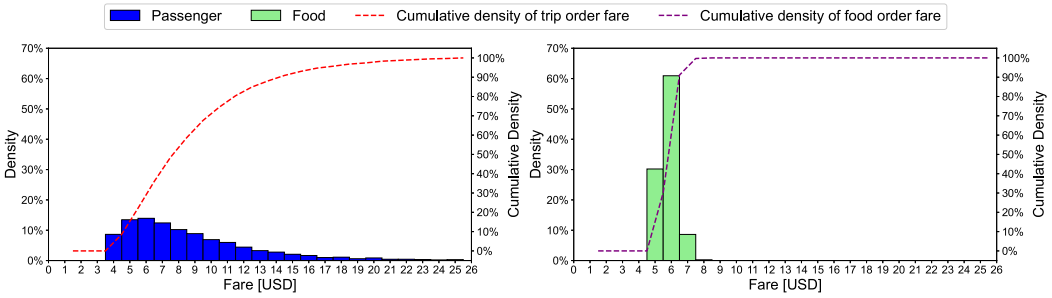


Fig. 6. Distributions of fares of trip and food orders.

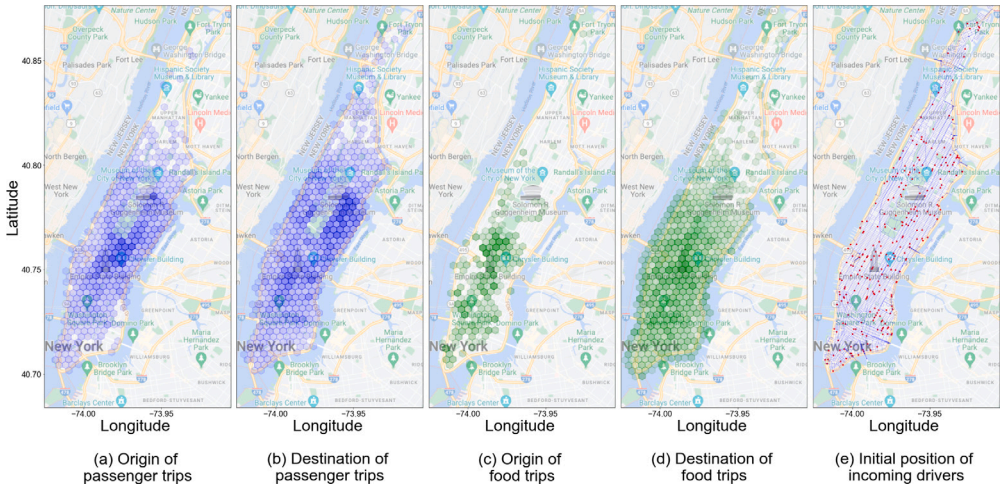


Fig. 7. Heatmap of origins and destinations of passenger and food orders in test data. Dark coloured hexagons indicate the most origins or destinations; light coloured hexagons indicate moderate origins or destinations, and other areas indicate minor or no origins or destinations. (e) corresponds to the distribution of the initial positions of incoming drivers in the road network. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

The new drivers' arriving rate is sampled from uniform discrete distributions with ranges between 1 and 5 [veh] per minute from 05:00 PM–06:00 PM. As depicted in Fig. 7(e), the initial positions (i.e., red dots) of arriving drivers are sparsely distributed all over the Manhattan network (i.e., blue links).

5.2. Simulation setup

Fig. 8 presents three phases in the simulation process with a ten-second rolling time window. In the first phase, **Assortment**, the demand, supply, and a set of priori known preference parameters are given to the menu assortment methods to determine personalized dispatch menus for each driver. We considered four benchmark methods stated in the following, including joint as well as disjoint menu assortment algorithms for comparison.

1. **Global menu**, lists all available (unmatched) orders in the menus of every idle driver.
2. **Local menu**, lists available (unmatched) orders in the menu of specific drivers who are within a circle centred at the order's origin. The radius of the circle is set to 3 [km].
3. **One-to-one MTPT**, is commonly employed in order dispatching literature by solving the maximum weight bipartite matching problem to minimize the total pick-up time (MTPT) of orders. Orders O and drivers D are abstracted as two sets of vertices, where each edge between driver d and order o has a weight of $1/\tau(l_d, l_o^{\text{org}})$. To reduce computational complexity, a dispatching radius (Yang et al., 2020a) is employed to eliminate edges whose pick-up distance exceeds 3 [km]. We employ the Kuhn–Munkres (KM) algorithm (Munkres, 1957) to solve the bipartite matching.
4. **One-to-one MTSP**, is the same as **One-to-one MTPT** except that its objective function is to maximize the total selection probability (MTSP) of drivers, i.e., the weight is set to $u_{d,o}/(u_{d,o} + u_d^{\text{c}})$ on the edge between driver d and order o.

According to the drivers' order acceptance behaviour experiments conducted in Ashkrof et al. (2021) and Hong et al. (2020), the assortment algorithms assume the priori known preference parameters of individual drivers are homogeneous with values selected

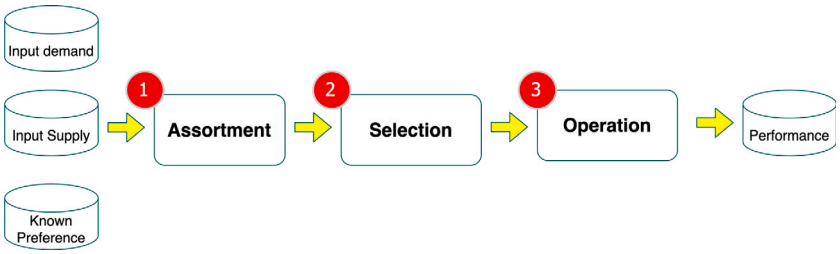


Fig. 8. Simulation phases of the experiments.

Table 1
The preference parameters of individual drivers in the **Selection** phase. μ, σ, a, and b denote mean, variance, minimum and maximum of the truncated Gaussian (TG) distributions, respectively.

Parameters	μ	σ	a	b
$\widetilde{\beta}_{0,d}$	0.0	0.5	−0.5	0.5
$\widetilde{\beta}_{1,d}$	2.0	0.5	1.5	2.5
$\widetilde{\beta}_{2,d}$	3.2	0.2	3.0	3.4
$\widetilde{\beta}_{3,d}$	0.6	0.1	0.5	0.7
$\widetilde{\beta}_{4,d}$	8.0	1.0	7.0	9.0
$\widetilde{u}_d^{\mathrm{c}}$	15.0	5.0	10.0	20.0

as $\beta_{0,d} = 0.0$, $\beta_{1,d} = 2.0$, $\beta_{2,d} = 3.2$, $\beta_{3,d} = 0.6$, $\beta_{4,d} = 8.0$, and $u_d^{\mathrm{c}} = 15.0$, $\forall d \in D$. $\beta_{0,d} = 0.0$ indicates the constant term is disregarded, $\beta_{1,d} = 2.0$ suggests the value of marginal preference of trip orders over food orders, $\beta_{2,d} = 3.2$ indicates the utility of the fare, $\beta_{3,d} = 0.6$ is the operational cost per minute, which considers the fuel cost and value of time for drivers (Association et al., 2012), and $\beta_{4,d} = 8.0$ is the preference for order destination, which is assumed to be the upper quartile of order fares. Given $\beta_{0,d}$, $\beta_{1,d}$, $\beta_{2,d}$, $\beta_{3,d}$, and $\beta_{4,d}$, we have investigated the average value of driver–order utilities and ultimately chose the rounded integer, 15, as *decline* utility ($u_d^{\mathrm{c}} = 15$, $\forall d \in D$) for each driver.

In the second phase (see Fig. 8), **Selection**, drivers' actual selections are determined by the individual driver's choice simulation. At the beginning of the simulation, each driver d, $d \in D$ is assigned a set of individual preference parameters $\widetilde{\beta}_{0,d}$, $\widetilde{\beta}_{1,d}$, $\widetilde{\beta}_{2,d}$, $\widetilde{\beta}_{3,d}$, $\widetilde{\beta}_{4,d}$, and $\widetilde{u_d}^{\mathrm{c}}$ sampled from truncated Gaussian (TG) distributions, see Table 1. Consequently, different drivers have distinct sets of individual preference parameters. At each dispatching window (every 10 s), once driver d receives a dispatch menu $O_d(X^*)$ from the **Assortment** phase, the utility of driver d for servicing order $o \in O_d(X^*)$, $\widetilde{u}_{d,o}$, is

$$\widetilde{u}_{d,o} = \overbrace{\widetilde{\beta}_{0,d}}^{\text{constant}} + \overbrace{\widetilde{\beta}_{1,d} \cdot I_o}^{\text{order type}} + \overbrace{\widetilde{\beta}_{2,d} \cdot f_o}^{\text{order fare}} - \overbrace{\widetilde{\beta}_{3,d} \cdot \tau(l_d, l_o^{\mathrm{org}})}^{\text{pickup time}} + \overbrace{\widetilde{\beta}_{4,d} \cdot V(l_o^{\mathrm{dest}})}^{\text{value of the order's destination}} . \tag{49}$$

Note that $\widetilde{\beta}_{0,d}$, $\widetilde{\beta}_{1,d}$, $\widetilde{\beta}_{2,d}$, $\widetilde{\beta}_{3,d}$, and $\widetilde{\beta}_{4,d}$ are individual preference parameters of driver d, which are different from the homogeneous preference parameters used by the platform in the **Assortment** phase. Following this, driver d making actual choices among orders in $O_d(X^*)$ using a logit probability, as:

$$\widetilde{p}_{d,o}(X^*) = \frac{\exp(\widetilde{u}_{d,o})}{\exp(\widetilde{u}_d^{\mathrm{c}}) + \sum_{o' \in O_d(X^*)} \exp(\widetilde{u}_{d,o'})}, \forall o \in O_d(X^*), \forall d \in D \tag{50}$$

where $\widetilde{u}_d^{\mathrm{c}}$ is also an individual preference parameter different from the homogeneous one used by the platform.

In the last phase, **Operation**, the platform collects the drivers' actual selections and assigns orders to the nearest responded driver. Assigned drivers head for the origins of orders to pick up the passenger or food, while idle drivers park at their current position to wait for the subsequent matching epoch (i.e., after 10 [s]). Occupied vehicles take the passenger or food to their destination by the shortest path, and then they will become idle after they drop off the passenger or the food order. Passengers or food requesters will cancel the order if not being matched within their matching patience. Individual drivers are also assumed to be impatient and will leave the system once they receive no match over 60 [min]. Finally, we report the system performance.

We use first two weekdays (i.e., 10 days) in December 2020 as the test data and report eight evaluation metrics:

1. **Avg. Match**. The average number of orders that are successfully served per test day.
2. **Avg. Cancellation**. The average number of orders that are cancelled per test day.
3. **Avg. Match time**. The average response time to orders from requesting (arriving in the network) to being matched.
4. **Avg. Pick-up time**. The average pick-up time of orders from being responded (matched) to being picked up. It is worth mentioning this is equal to the average deadheading time for vehicles.
5. **Avg. Occupied rate**. The average occupied rate per vehicle, defined as the ratio of the time spent on serving orders to the total operating time of the vehicle.

Table 2
The results of different assortment methods during 05:30 pm–09:00 pm (averaged over 10 test days). The numbers in the parentheses are the average response rate and the average cancellation rate of the platform. Note that the sum of the response and cancellation rates might be less than 100 percent. This is because some orders may still wait in the system at 09:00 pm.

Methods	Avg. match	Avg. cancellation	Avg. match time [s]	Avg. pick-up time [s]	Avg. occupied rate	Avg. leaving vehicles	Daily driver revenue		Avg. CPU time [s]
							Mean [USD]	Std [USD]	
Global menu	4081.0 (65.8%)	2113.0 (34.1%)	47.8	206.8	61.2%	22.6	129.7	41.3	0.0
Local menu	4194.0 (67.6%)	1973.0 (31.8%)	50.8	190.3	66.5%	58.3	130.8	51.9	0.0
One-to-one MTPT	4429.0 (71.4%)	1716.0 (27.7%)	57.4	93.5	67.6%	100.3	120.5	72.0	0.1
One-to-one MTSP	4453.0 (71.8%)	1724.0 (27.8%)	38.2	197.6	66.4%	43.8	136.8	47.5	0.1
SG for disjoint menu	4506.0 (72.6%)	1696.0 (27.3%)	35.7	277.7	61.0%	15.1	139.5	18.5	0.9
LS for joint menu	4693.0 (75.7%)	1510.0 (24.3%)	28.3	253.9	62.6%	15.8	143.4	20.7	7.8
γ^*-greedy for joint menu	4542.0 (73.2%)	1660.0 (26.8%)	34.2	265.0	61.2%	16.2	140.0	17.9	0.1

6. **Avg. Leaving vehicles**. The average number of leaving vehicles per test day.
7. **Daily driver revenue**. The mean and standard deviation (Std) of daily revenue of individual drivers within testing days.
8. **Avg. CPU time**. The average calculation time of the menu assortment per optimization interval (i.e., 10 [s]) per test day.

From the perspective of the platform operator, Avg. Match, Avg. Cancellation, and Avg. Match time are the measurements of system efficiency. Avg. Leaving vehicles measures the ability to maintain the drivers' participation in the system. Avg. Pick-up time and Avg. Occupied rate measure the satisfaction levels of customers and drivers, respectively. Assume the amount that the platform pays the drivers is 100% percentage of the total paid by customers, Daily individual revenue assesses monetary earnings per day for drivers in the system. Avg. CPU time measures the computation time of the menu assortment methods. The experiments are conducted on a MacBook Pro with an Apple M1 chip and 8 GB RAM.

5.3. Dispatch menu algorithms performance

Table 2 summarizes the numerical results of the seven menu assortment methods, where parameter $\lambda = 0.001$ is fine-tuned based on multiple rounds of testing (see Appendix C. Parameter tuning) for the LS algorithm (Algorithm 3). LS for the joint menu achieves the highest number of matches, the lowest number of cancellations, and the shortest match time. γ^*-greedy for joint menu (Algorithm 1) provides the second most matches, the second least number of cancellations, and the second shortest match time. However, it can be observed that the averaged CPU time of the LS algorithm scales up dramatically to 7.8 [s], due to the exponential nature of the algorithm. In contrast, the γ^*-greedy algorithm performs efficiently with a computation time of 0.1 [s]. The average numbers of matches and cancellations, and the average match time of the SG algorithm are also better than those of the other four benchmark strategies. It is evident from the results that the proposed menu assortment methods can boost the system efficiency by offering the dispatch menus to individual drivers. Offering a menu for the drivers, however, might have a number of side effects like increasing the order pick-up time. However, Table 2 shows that the pick-up time of the passengers only increases within reasonable bounds (less than 5 [min]). This acceptable increase can be explained by the fact that the passenger pick-up time is the factor partially considered in the driver's choice behaviour (see Eq. (49)).

It is also worth noticing that one-to-one MTPT method yields a significantly shorter pick-up time than other methods do. This is expected, as one-to-one MTPT method involves a customer-centric objective to minimize the pick-up time of all the matched orders. In contrast, one-to-one MTSP method integrates driver-centric and customer-centric goals simultaneously. As a result, one-to-one MTSP method can achieve more promising results in terms of platform efficiency and customers' and drivers' satisfaction. However, both methods raise the issue of reducing driver participation in the system. This is because the total number of drivers with non-empty menus is limited. This low driver participation discourages long-term suppliers' loyalty. As a result, the average numbers of leaving vehicles of one-to-one MTPT and MTSP methods explode to 100.3 [veh.] and 43.8 [veh.] per test day, respectively.

Fig. 9 displays the average number of drivers' selections and the average number of drivers with no dispatching (an empty menu) per ten seconds over testing days. There are three types of outcome for drivers' selections, (i) matching (i.e., green), the selection is matched after the platform's assignment, (ii) duplicate (i.e., blue), the selection is unmatched since at least another driver has chosen the same order, and (iii) decline (i.e., red), the driver declines the dispatch menu. A noticeable observation in Fig. 9 is that the global menu achieves the most selections (i.e., the sum of blue and green). This is because increasing the number of orders presented into the menus of individual drivers increases the chances of finding an acceptable order instead of declining. It can be also seen that the number of duplicates rises dramatically and reaches high proportions of total selections. This is because the global menu method inherently creates collisions, i.e., drivers will select only a few high-value (high-utility) orders. Fig. 9 suggests that the global menu method leads to the highest collision cost in the menu assortment, which deteriorates the system efficiency and the experience of customers and drivers. To hedge against the collision cost, one-to-one MTPT and MTSP methods list each order into one dispatch menu and also limit the menu size to one. Evidently, both methods lead to no duplicates of driver selections (see Fig. 9). It can be seen in Fig. 9 that the number of drivers with no dispatching accounts for a significant proportion of the total available supply, while the number of selections is scarce over testing hours.

Fig. 10 shows the average numbers of waiting (unmatched) orders and idle drivers before the execution of dispatching in the market. Note that the sum of the numbers of no dispatching, duplicate, matching, and decline in Fig. 9 is the number of idle drivers (i.e., red line) in Fig. 10. The matching friction between customers and drivers can be observed in global menu, local menu, one-to-one MTPT, and one-to-one MTSP methods: large numbers of waiting orders and idle drivers co-exist in the market. This is due to the collisions in global menu and local menu methods, and the low driver participation (dispatch acceptance) in one-to-one MTPT and one-to-one MTSP methods. In contrast, the uniform distributions of waiting orders and idle drivers can be observed in SG, LS, and γ^*-greedy algorithms, which shows a period (05:30 PM to 08:00 PM) of maximizing supply utilization, and a period (08:00 PM to 09:00 PM) of achieving the highest order fulfilment. This demonstrates that the proposed three menu assortment methods significantly reduce the matching friction in the market. Fig. 11 displays the average and cumulative numbers of drivers who left the market. A notable feature in this figure is that drivers continuously leave the market from 05:30 PM to 08:30 PM with local menu, one-to-one MTPT, and one-to-one MTSP methods, which provides evidence for the previous claim that these three methods would avert drivers (supply) from the market. In contrast, only a small percentage of drivers (less than 20) start to leave the market only after 8:30 PM with our proposed menu-based algorithms. This demonstrates the effectiveness of our proposed algorithms in achieving higher driver retention compared to the local menu, one-to-one MTPT, and one-to-one MTSP methods. Fig. 12 presents the average numbers of active drivers (including idle and occupied drivers) in the market. It can be seen that the numbers of drivers matched and occupied with trip orders are significantly higher than that matched and occupied with food orders, which is consistent with the assumption of the drivers' choice preference in Table 1.

Fig. 13 presents the distribution of the daily revenue of each driver with different menu assortment methods. The standard deviations (see Table 2) of the daily revenue of the global menu, local menu, one-to-one MTPT, and one-to-one MTSP methods are higher than 40 [USD]. These considerable revenue discrepancies among individual drivers are due to the concentrative choices (i.e., selecting high-utility orders) of drivers in the global menu and local menu methods, and the restrictive dispatching rules (i.e., the menu size of at most 1 for each driver) of one-to-one MTPT and one-to-one MTSP methods. Such menu assortment methods lead to high-income inequality and are unfair to individual drivers. Consequently, the average numbers of leaving vehicles of these four methods are greater than 30 in Table 2. In contrast, the average driver revenue of the LS algorithm is higher than that of other methods. The γ^*-greedy and SG algorithms can also achieve promising and fair individual revenue distributions.

Fig. 14 shows the distributions of the menu size of different menu assortment methods. It shows the dispatch menus of global menu and local menu methods contain approximately 55.7 orders and 3.8 orders on average. It also seems evident that one-to-one MTPT and MTSP yield the smallest menu size, and most empty dispatch menus (0 or 1). In contrast, the three proposed menu assortment algorithms allow flexible menu sizes for each driver. Although there is no restraint on the maximum size of the dispatch menu (i.e., $b = \infty$ in Eq. (7)), dispatch menus of the SG and LS algorithms can be narrowed down to a size of 1.9 and 0.6 orders on average, respectively. γ^*-greedy algorithm enlarges the menu size to allow the drivers to view on average 10.0 orders in their dispatch menu.

5.4. Extension to meal delivery bundling problem

In this section, a new experiment is conducted with the identical parameter setting and dataset to the previous experiment in Section 5.3 to further investigate the performance of the proposed menu assortment methods with bundling meal orders. Practically to better utilize supply capacity considering the reality of a limited number of suppliers in the market, meal orders can be bundled where a single driver handles multiple orders in a single trip. When bundling order is considered, the proposed menu assortment problem can be divided into smaller, more manageable sub-problems as follow.

(i) Two-step cluster-based bundling: The basic idea is to prioritize the grouping of restaurants before meal orders. The approach begins by creating a hexagonal grid over the network, effectively dividing the area into discrete spatial units. Each restaurant is then assigned to the hexagon that encompasses its location, forming distinct restaurant groups based on spatial closeness. Once the restaurant groups are obtained, the delivery orders are extracted for each restaurant group in every matching instance. The fundamental idea of the next step is to create spatial-based clusters of meal orders with the same restaurant group. For every restaurant group, the collected orders are bundled by Algorithm 4. Appendix D. Cluster-based bundling algorithm provides a detailed breakdown of Algorithm 4 to consolidate multiple orders into one route.

Fig. 15 showcases an example of the cluster-based bundling approach. Fig. 15(a) presents the hexagonal grids generated over Manhattan, covering the locations of restaurants. Each restaurant (coloured circle point) is assigned to the hexagon that encompasses its location, establishing the initial grouping of restaurants within each hexagon. Fig. 15(b) displays a bundling example of one selected restaurant group (hexagon), where each grey circle within the hexagon depicts a restaurant, and each square point represents a customer and is colour-coded according to the assigned bundle (cluster). This example identifies five distinct bundles for the restaurant. The bundles exhibit clear boundaries, indicating distinct geographic proximity patterns among order destinations. It is noteworthy that bundle 5 has two orders requiring multiple pickups and drop-offs when the courier carries out deliveries.

(ii) Menu assortment optimization: Once the orders have been clustered into bundles, each bundle can be viewed as an item in the driver menu. Then menu assortment optimization is required to determine which bundles should be listed in the menu of which driver. Each driver-bundle pair (d, s) has a specific utility function for driver d,

$$u_{d,s} = \overbrace{\beta_{0,d}}^{\text{constant}} + \overbrace{\beta_{1,d} \cdot I_s}^{\text{type}} + \overbrace{\beta_{2,d} \cdot f_s}^{\text{bundle fare}} - \overbrace{\beta_{3,d} \cdot \tau(l_d, l_s^{\text{org}})}^{\text{pickup time to the last restaurant}} + \overbrace{\beta_{4,d} \cdot V(l_s^{\text{dest}})}^{\text{value of the last destination}} \tag{51}$$

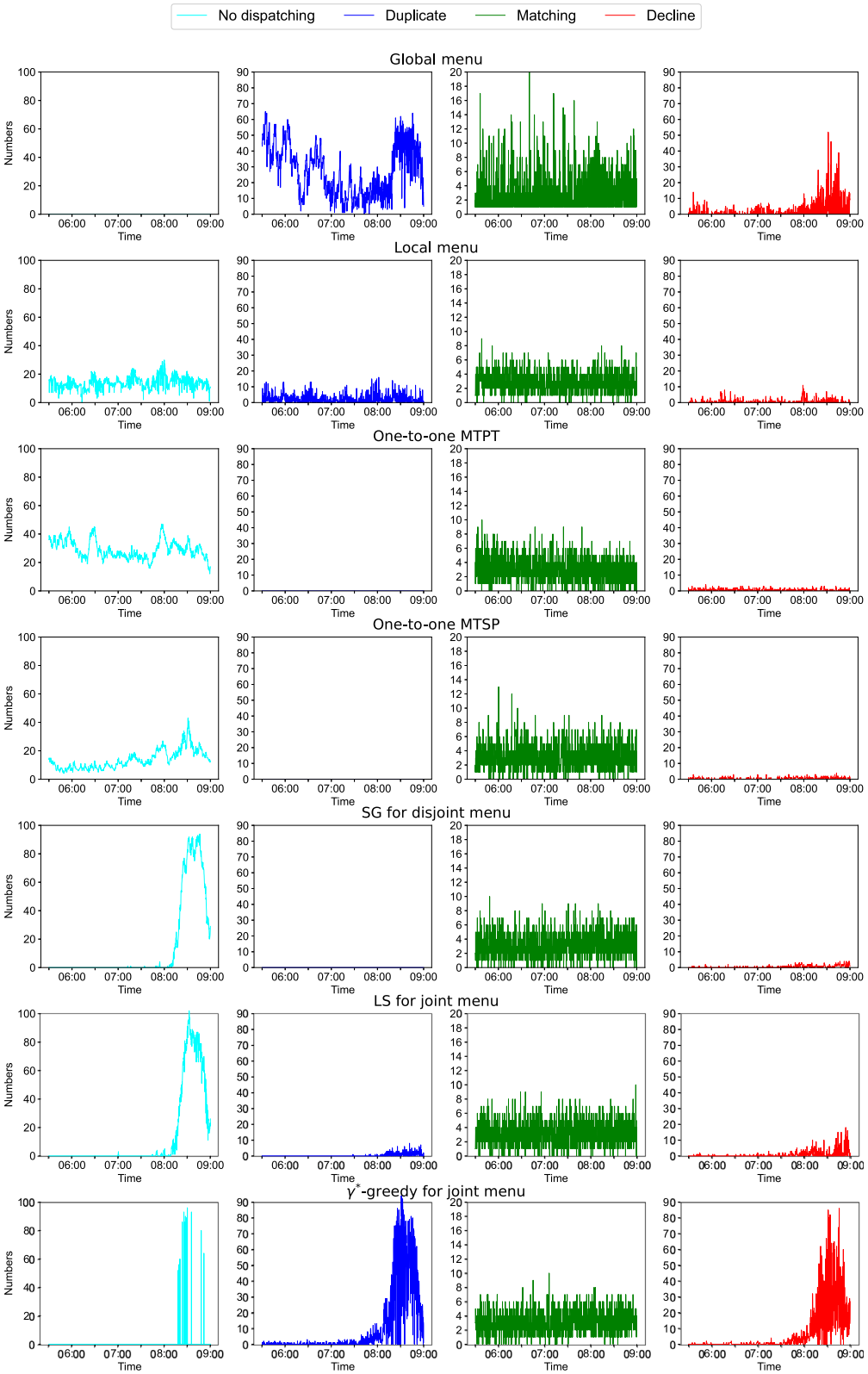


Fig. 9. Average number of drivers' selections at each matching instance (10 [s]) within 10 testing days.

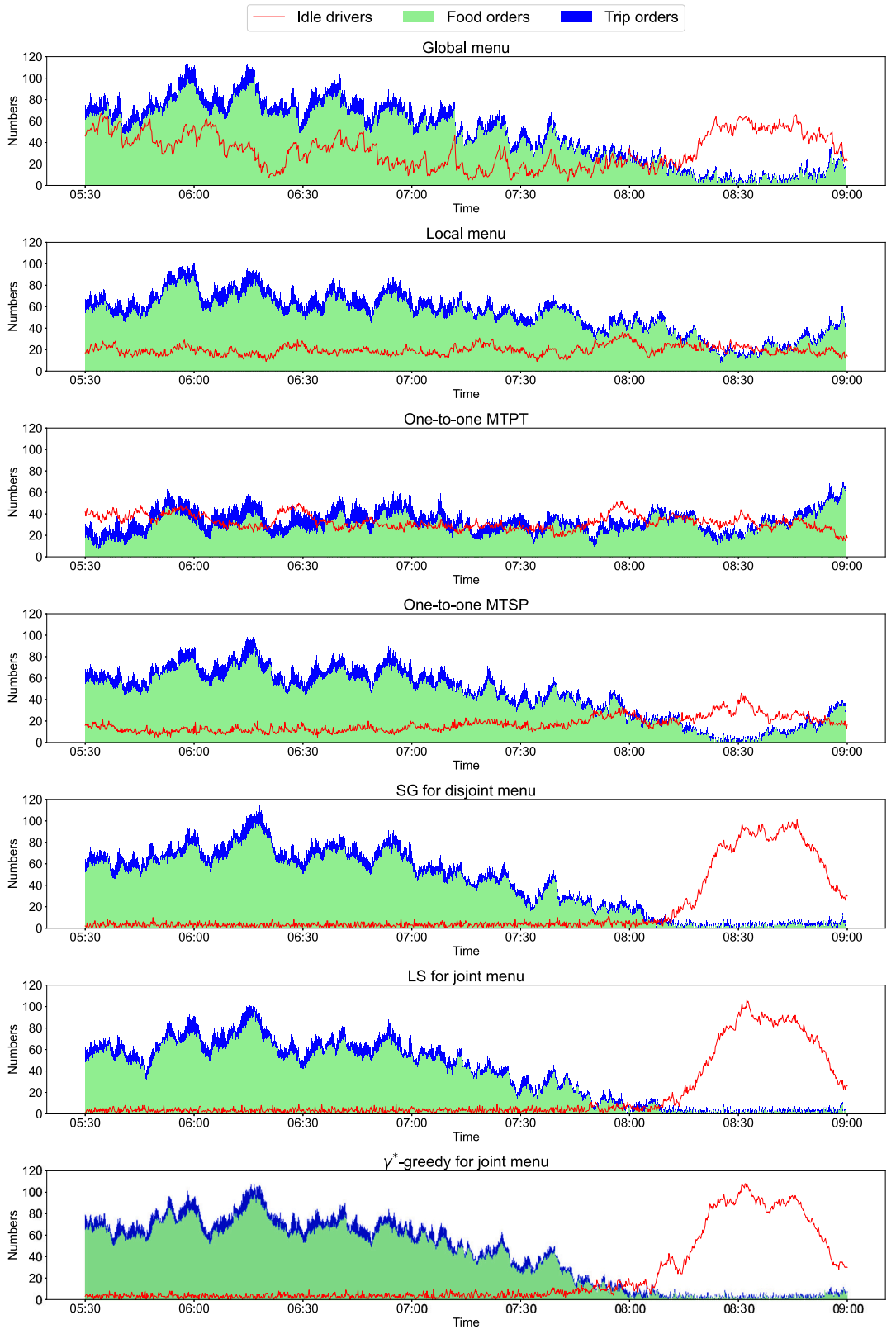


Fig. 10. Average numbers of waiting orders (passenger and food) and available idle drivers at each matching instance (10 [s]) within 10 testing days. Note the matching friction (coexistence of waiting orders and idle vehicles) with global menu, local menu, one-to-one MTPT, and one-to-one MTSP methods. The matching friction exists because of autonomy of drivers to not accept the platform dispatch orders. The menu assortment methods (joint and disjoint) curb this friction.

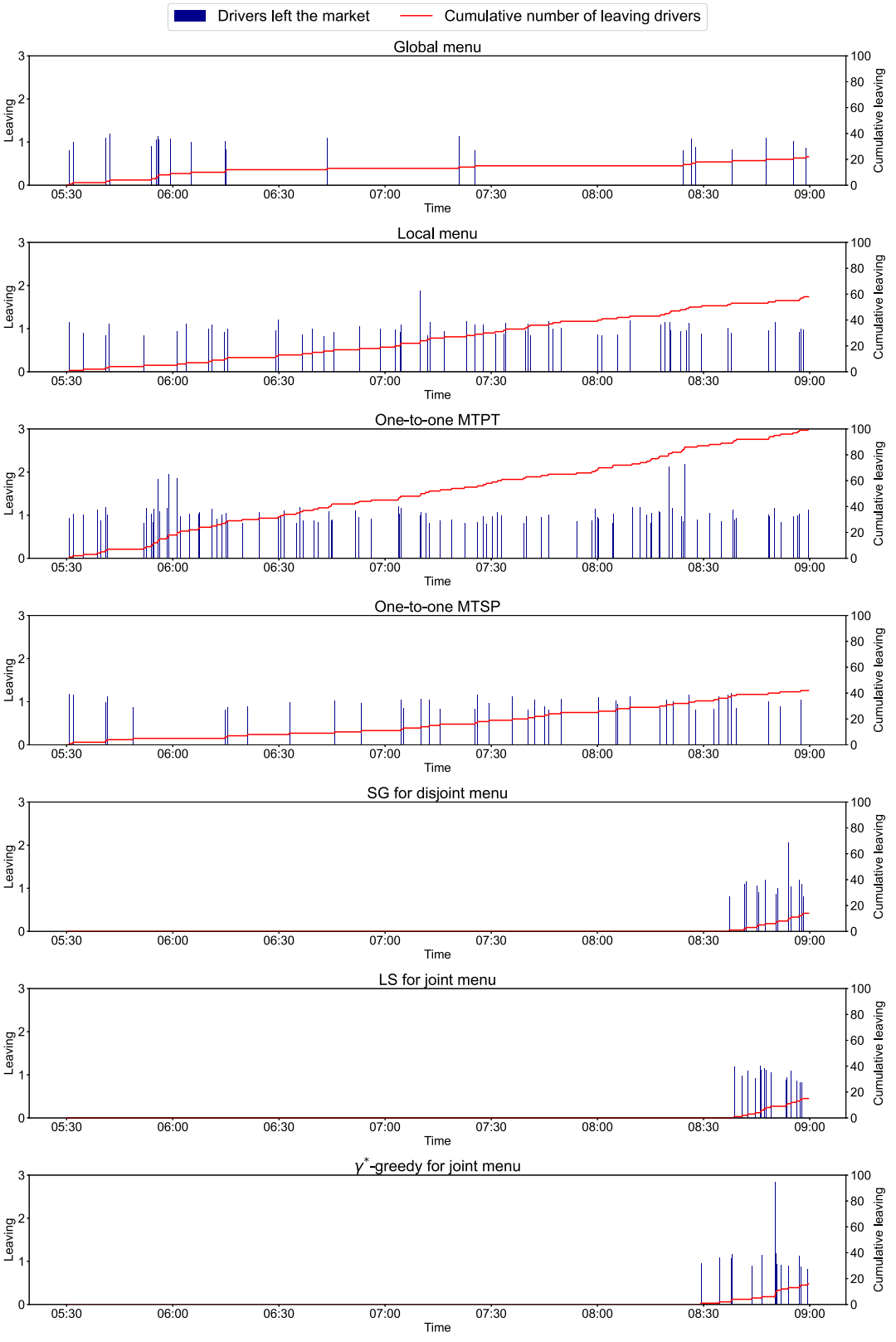


Fig. 11. Average number of drivers who left the market at each matching instance (10 [s]) over the 10 testing days. The second y-axis and curve depict the cumulative number of leaving vehicles.

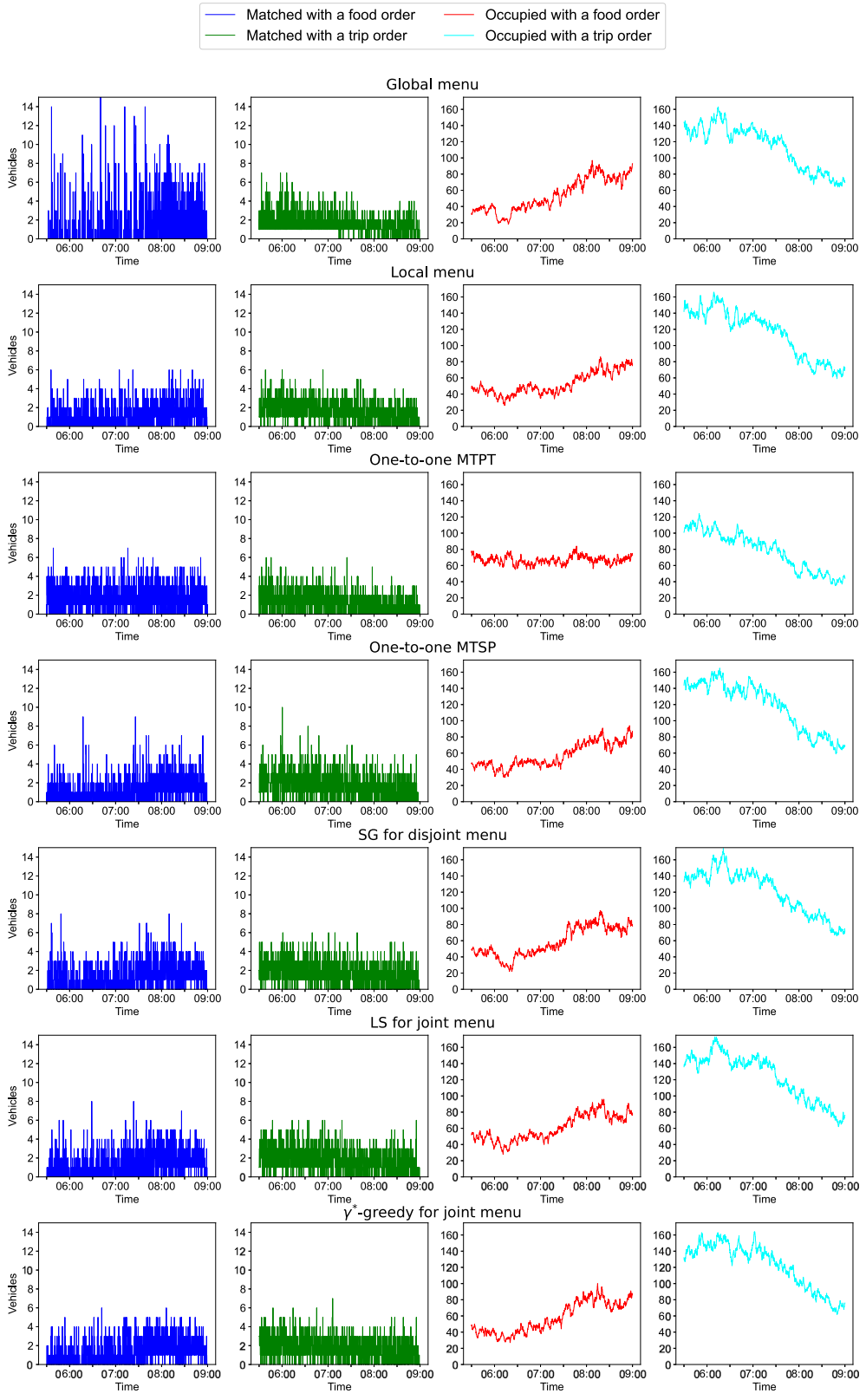


Fig. 12. Average number of drivers at each matching instance (10 [s]) matched and occupied with food and passenger orders within 10 testing days.

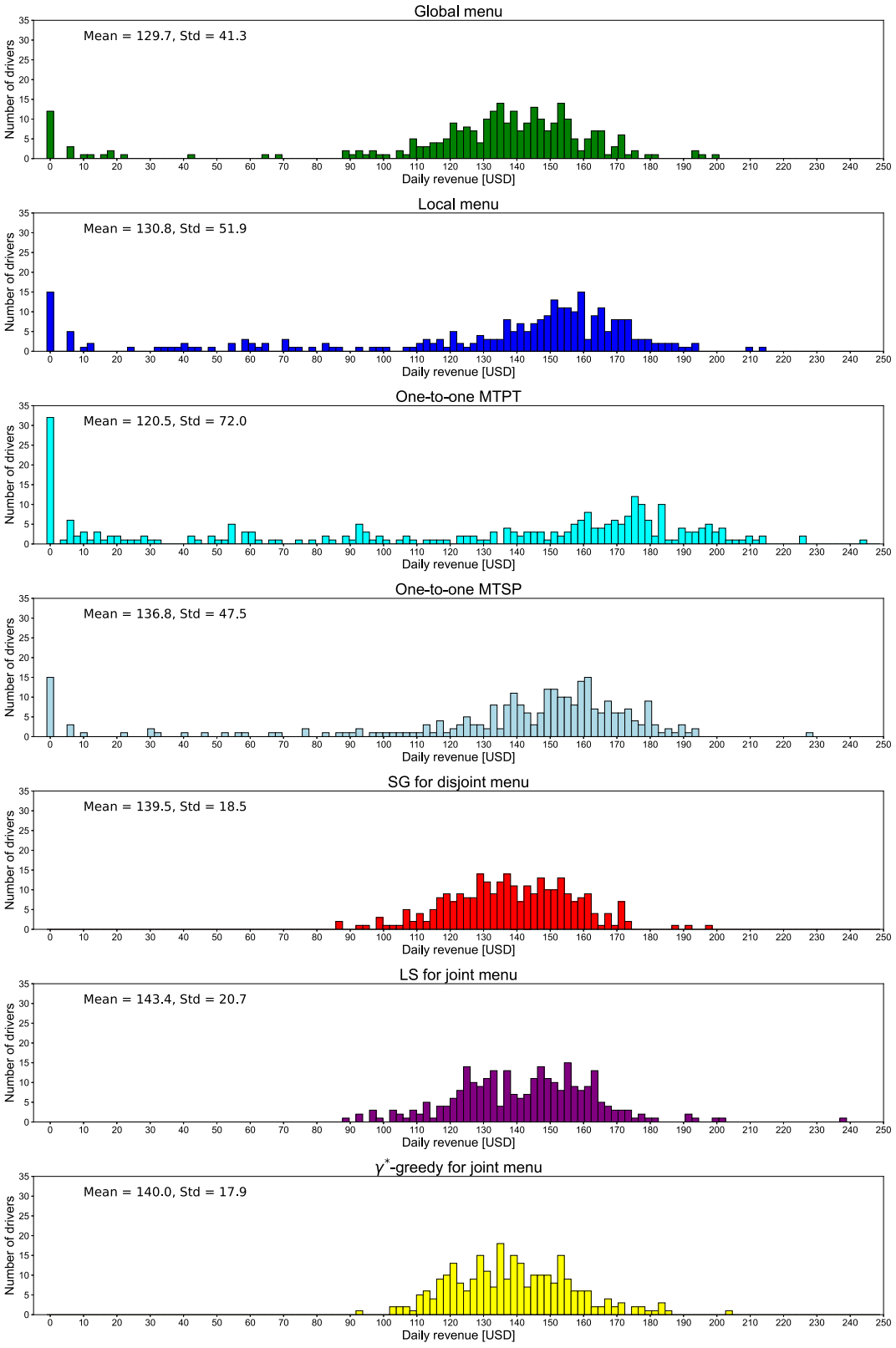


Fig. 13. Average distributions of daily revenue for each driver with the seven assortment methods. The bin width is 2 [USD].

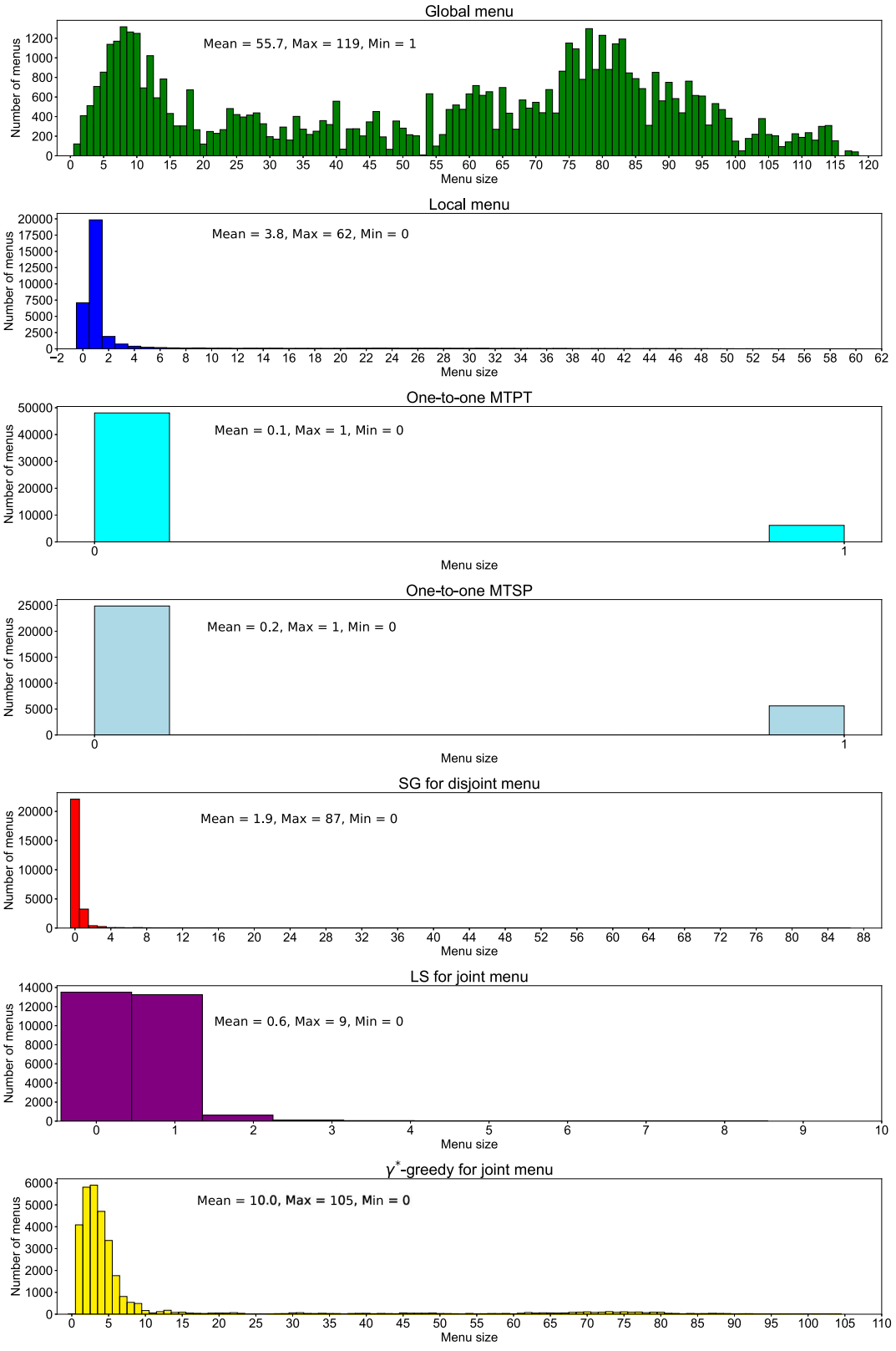


Fig. 14. Distributions of menu sizes of the seven assortment methods. The limits of x and y axes are different for visual purposes.

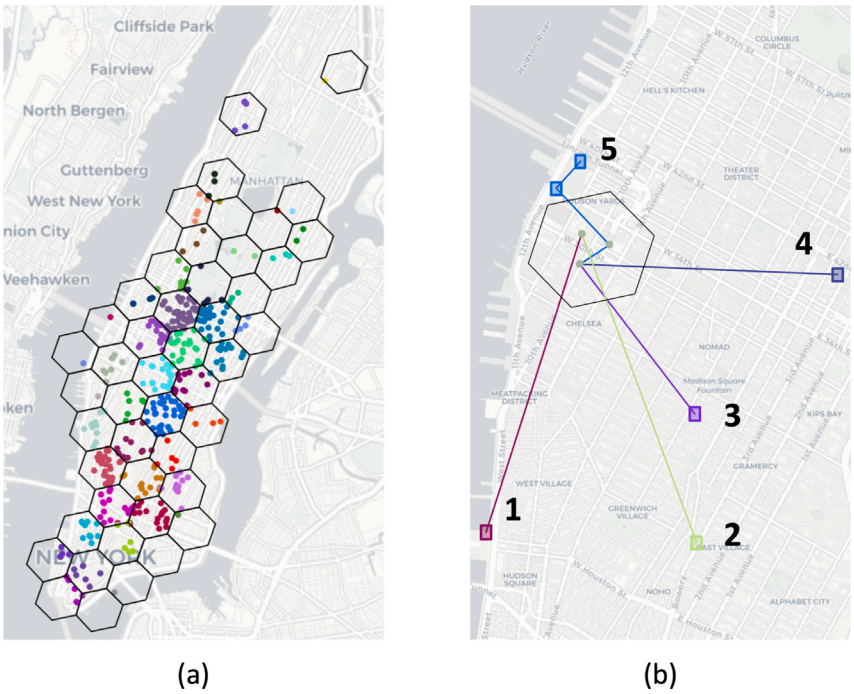


Fig. 15. Example of bundling meal orders for a specific restaurant group (hexagon). (a) groups the restaurants across different hexagonal grids. Each data point represents a restaurant, and the groups are distinguished by different colours. (b) visualizes five calculated bundles (clusters) in the example hexagon.

where f_s represents the total fare charged on bundle s, which is assumed to be the sum of order fares (see Eq. (57)) in this bundle. Note that routing problems are determined independently for the pick-up and drop-off stages of each bundle s. Travelling salesman problem (TSP) are firstly used to optimize the pick-up sequences (restaurants) in bundle s, and then to determine the optimal sequences of drop-off locations in bundle s. Both objectives of the two stages are set to minimize the total distance travelled. Hence, $\tau(l_d, l_s^{\text{org}})$ is the pick-up time from driver d's current location, l_d, to the last visited restaurant of bundle s, l_s^{org}. $V(l_s^{\text{dest}})$ is the spatial value function of the last drop-off location of bundle s, l_s^{dest}. Without loss of generality, the value of I_s is assumed to be -1. This assumption is justified based on the fact that bundle s might have more pick-ups and drop-offs than a single meal order. Hence, we account for a higher service time uncertainty in the bundling delivery problem. Consequently, the problem can be formulated as Eqs. (5) to (7).

The experimental setup involved carefully selecting and manipulating parameters in the proposed bundling algorithm. Each hexagon is assigned a unique index based on its resolution level in Uber H3 system,[2] and the average edge length of each hexagon is set to 0.53 [km]. The distance radius ρ is set to 0.5 [km], reflecting the maximum distance between two arbitrary orders in the same bundle (trip). MaxPts is set to 4 to represent the driver capacity for meal delivery. Consequently, two types (i.e., single passenger trip and meal bundled trip) of menu items are considered in the menu assortment problem to measure the performance of the proposed approaches.

Table 3 summarizes the numerical results of the seven menu assortment methods with the same parameter setting as Table 2. In line with Table 2, Table 3 reveals that the proposed menu assortment methods (i.e., SG, LS, and γ^*-greedy) achieve higher numbers of matches, lower numbers of cancellations, and shorter match times compared other benchmark methods. This similar observation in both experiments reinforces the validity and reliability of the previous findings, as it demonstrates the consistency and robustness of the effects across different experimental conditions and settings. Another interesting observation is that the results of Table 3 surpass the performance observed in Table 2 (e.g., higher matching rate, shorter matching time, and higher occupied rate), indicating significant advancement in platform efficiency, and customers' and drivers' experience. This finding highlights the compelling advantages of order bundling in meal delivery operations.

Fig. 16 shows the average numbers of waiting passengers, meal bundles, and idle drivers before the execution of menu dispatching in the market. Despite efforts to consolidate multiple meal orders into bundles, a significant majority of the formed bundles exhibit a single-order composition (i.e. bundles with one order). The prevalence of single-order bundles can be attributed to the value of ρ and the demand intensity in the market. While some nuanced differences emerged in the magnitude or specific menu assortment

[2] https://h3geo.org/.

Table 3
The results of different assortment methods with bundling meal orders during 05:30 pm–09:00 pm (averaged over 10 test days).

Methods	Avg. match	Avg. cancellation	Avg. match time [s]	Avg. pick-up time [s]	Avg. occupied rate	Avg. leaving vehicles	Daily driver revenue		Avg. CPU time [s]
							Mean [USD]	Std [USD]	
Global menu	4315.0 (70.7%)	1791.0 (29.3%)	30.9	199.6	67.0%	20.6	70.8	34.8	0.0
Local menu	4679.0 (76.6%)	1425.0 (23.3%)	29.7	197.0	73.4%	46.3	76.1	39.3	0.0
One-to-one MTPT	4593.0 (75.2%)	1496.0 (24.5%)	52.4	102.8	76.8%	77.3	64.3	41.5	0.1
One-to-one MTSP	4626.0 (75.8%)	1477.0 (24.2%)	35.8	174.8	75.1%	55.8	73.2	40.0	0.1
SG for disjoint menu	4722.0 (77.3%)	1383.0 (22.6%)	21.9	330.3	65.7%	12.1	80.0	24.9	0.7
LS for joint menu	4931.0 (80.8%)	1175.0 (19.2%)	21.7	276.9	68.2%	13.8	84.7	27.0	6.5
γ^*-greedy for joint menu	4724.0 (77.4%)	1382.0 (22.6%)	21.4	316.3	66.2%	8.2	80.4	26.6	0.2

methods, the overall trend of matching frictions in Fig. 16 closely aligns with that in Fig. 10, indicating a strong level of consistency and validity.

6. Summary and future work

In traditional on-demand mobility markets, suppliers work as full-time employees of the platform and have to follow the order dispatching instructions given by the platform. The prosperity of sharing economy attracts more freelance suppliers to join the market to share their transportation assets (e.g., vehicles). Consequently, the platform faces the challenge of dispatching request orders to freelance suppliers; that is, the freelance suppliers serve their self-interest and might decline the instructions from the platform according to their preferences.

This paper has proposed novel menu assortment approaches to provide freelance drivers a personalized dispatch menu to choose from. The approaches address the heterogeneity and autonomy of suppliers to choose among several dispatch orders and to have a *decline* option. The choice behaviour of suppliers is modelled probabilistically by considering their preferences for attributes such as order type, fare, pick-up distance, and destination. We introduced two menu assortment problems: disjoint menus, where the suppliers are assigned disjoint sets of orders, and joint menus, where the suppliers may share the same orders in their menus. The analysis shows that the objective function of the disjoint model is monotone non-decreasing submodular, while the objective function of the joint model is non-monotone non-submodular. We present a standard greedy (SG) algorithm to solve the disjoint assortment problem, and γ^*-greedy and local search (LS) algorithms for the joint assortment problem. Specifically, SG and LS can solve disjoint and joint menu assortment with approximation ratios of $1/2$ and $1/(3+\varepsilon)$ (ε is an arbitrary non-negative number) in the worst case, respectively. To maximize the utilization of delivery resources, this paper extends the proposed methods for meal delivery services by consolidating multiple orders into one route.

Compared to traditional dispatching policies applied in practice, the proposed menu assortment method can reduce the matching friction in the market according to our numerical results. The proposed methods can avoid the substantial simultaneous coexistence of idle drivers and unmatched orders. In addition, the results suggests that the proposed algorithms can improve platform efficiency, enhance customer experience, and provides benefits for suppliers by increasing their average daily income while increasing the income equity among the suppliers.

Various extensions can be explored in the future. Suppliers' order acceptance behaviour could change time by time, so it is insightful to investigate this behaviour by capturing more dynamic aspects (e.g., arriving and leaving of suppliers) of the market and considering more characteristics (e.g., multi-homing and working shift) of suppliers. Incorporating contextual factors such as real-time demand conditions, surge pricing, and working hours (driver fatigue), one promising research direction is the development of real-time prediction models that leverage streaming data and advanced learning techniques to forecast driver *decline* behaviour and utility on a per-order basis. Another research direction is to allow suppliers to select multiple orders; this could improve system performance, yet requires a more complicated menu assortment approach to solve the problem. Further, multiple ride-hailing platforms may co-exist and compete with each other in many local markets. The presence of inter-platform competition (Zhang and Nie, 2021) and multi-homing passengers and drivers also impose a layer of complexity for the menu assortment problem.

CRediT authorship contribution statement

Yue Yang: Conceptualization, Formal analysis, Investigation, Methodology, Software, Writing – original draft, Writing – review & editing. **Seeun William Umboh:** Conceptualization, Formal analysis, Investigation, Methodology, Validation, Writing – original draft. **Mohsen Ramezani:** Conceptualization, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Supervision, Validation, Writing – original draft, Writing – review & editing.

Acknowledgement

This research was partially funded by the Australian Research Council (ARC) Discovery Early Career Researcher Award (DECRA), Australia DE210100602.

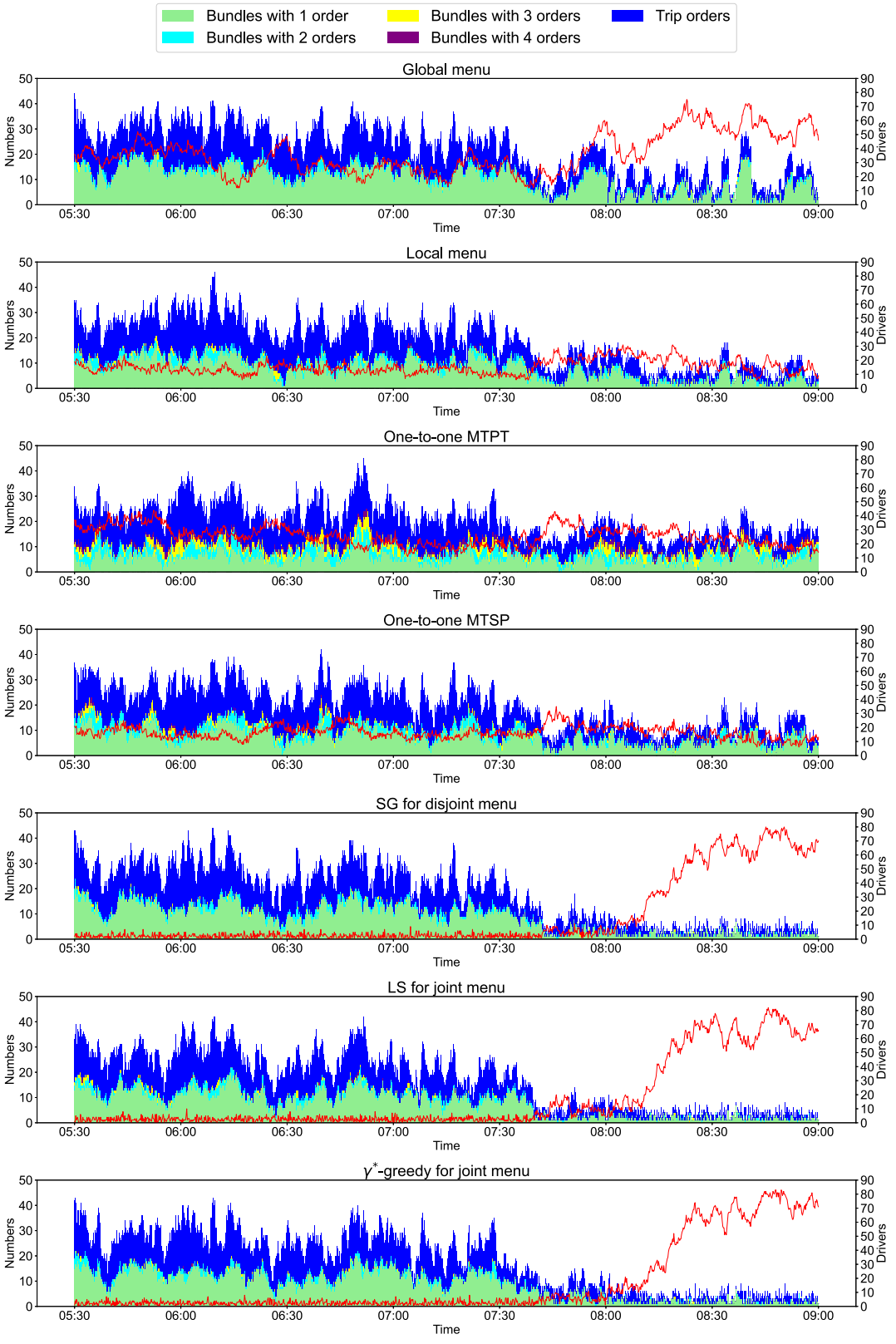


Fig. 16. Average numbers of waiting passengers, meal bundles, and available drivers at each matching instance (10 [s]) within ten testing days. The first y-axis and histogram reflect the number of passengers and bundles, and the second y-axis and curve depict the number of available drivers.

Appendix A. Mathematical preliminaries

This paper investigates the characteristics of the disjoint and joint menu assortment problems, either explicitly or implicitly, involving the concept of submodularity. Submodularity reflects a diminishing returns property for a set function, stating that adding an element to a smaller set increases the function value more than adding it to a larger set. For the sake of completeness, we start by introducing the following definitions:

Definition 2 (*Submodularity*)**.** A function $f : 2^{\mathbb{X}} \to \mathbb{R}$ is submodular if for any two sets $S \subseteq T \subseteq \mathbb{X}$, and an element x, $x \in \mathbb{X} \backslash T$,

$$f(S \cup \{x\}) - f(S) \geq f(T \cup \{x\}) - f(T) \tag{52}$$

where $\mathbb{X}$ is a ground set of elements, that is, $\mathbb{X} = D \times O$.

Definition 3 (*Supermodularity*)**.** A function $f : 2^{\mathbb{X}} \to \mathbb{R}$ is supermodular if for any two sets $S \subseteq T \subseteq \mathbb{X}$, and an element x, $x \in \mathbb{X} \backslash T$,

$$f(S \cup \{x\}) - f(S) \leq f(T \cup \{x\}) - f(T). \tag{53}$$

Definition 4 (*Matroid*)**.** A matroid M is a pair $(\mathbb{X}, I)$ where I is a collection of subsets of $\mathbb{X}$ (that we call 'independent'), satisfying two axioms:

- for any set $S \subseteq \mathbb{X}$ it must hold that $S \in I$, and for any set $T \in S$ it must hold that $T \in I$.
- for any sets $S, T \in I$ and $|T| \leq |S|$, it must hold that there exists an element $s \in S \backslash T$ such that $T \cup \{s\} \in I$.

In this paper, we are interested in two specific types of matroid: (i) uniform matroid, a uniform matroid is a matroid $(\mathbb{X}, I)$ such that for a positive integer k, $I = \{S : S \subseteq \mathbb{X}, |S| \leq k\}$. Thus, the uniform matroid only constrains the cardinality of the feasible sets in I. (ii) Partition matroid, a partition matroid is a matroid $(\mathbb{X}, I)$ such that for a positive integer n, disjoint sets $\mathbb{X}_1, \mathbb{X}_2, \ldots, \mathbb{X}_n$ and positive integers $k_1, k_2, \ldots, k_n$, $\mathbb{X} = \mathbb{X}_1 \cup \mathbb{X}_2 \cup \cdots \cup \mathbb{X}_n$ and $I = \{S : S \subseteq \mathbb{X}, |S \cap \mathbb{X}_i| \leq k_i, \forall i = 1, \ldots, n\}$.

Definition 5 (*Curvature*)**.** Consider matroid I for $\mathbb{X}$, and a nondecreasing submodular set function f: $2^{\mathbb{X}} \to \mathbb{R}$ such that (without loss of generality) for any element $s \in S$, $f(s) \neq 0$, and $S \subseteq \mathbb{X}$. The curvature measures how far f is from submodularity or linearity. Define curvature of f over matroid I as:

$$\mathsf{c}_f = 1 - \min_{s \in S, S \in I} \frac{f(S) - f(S \backslash \{s\})}{f(s)} \tag{54}$$

Note that the definition of curvature c_f implies that $0 \leq \mathsf{c}_f \leq 1$. Specifically, if $\mathsf{c}_f = 0$, it means for all the feasible sets $S \subseteq \mathbb{X}$, $f(S) = \sum_{s \in S} f(s)$. In this case, f is a linear (modular) function. In contrast, if $\mathsf{c}_f = 1$, then there exist a feasible $S \in I$ and an element $s \in X$ such that $f(S) = f(S \backslash \{s\})$. In this case, element s is redundant for the contribution of the value of f given the set $S \backslash \{s\}$.

Definition 6 (*Inverse Generalized Curvature*)**.** Given two subsets S and T that S, $T \subseteq \mathbb{X}$, and element s, $s \in S \backslash T$, the inverse generalized curvature (Bogunovic et al., 2018) of a non-negative function f is the smallest scalar $\check{c}_f \in [0, 1)$ such that

$$(1 - \check{c}_f) \leq \frac{f(S) - f(S \backslash \{s\})}{f(S \cup T) - f(S \cup T \backslash \{s\})}, \forall S, T \subseteq \mathbb{X}, s \in S \backslash T \tag{55}$$

and

$$\check{c}_f = 1 - \min_{\forall S, T \subseteq \mathbb{X}, s \in S \backslash T} \frac{f(S) - f(S \backslash \{s\})}{f(S \cup T) - f(S \cup T \backslash \{s\})}. \tag{56}$$

The function $f(\cdot)$ is submodular if $\check{c}_f = 0$.

Appendix B. Synthesizing food orders

To examine the effectiveness of the proposed methods to address drivers heterogeneity in servicing different type of orders, the numerical experiments include a set of synthetic food orders. We consider 447 real-world restaurants in Manhattan in this process. Specifically, the arrival of food orders is assumed to follow a Poisson distribution with a rate of 3 orders per ten seconds. The restaurant of each order is sampled probabilistically according to their normalized rating score.

The delivery distance of each order is sampled from a fitted gamma distribution (see Fig. 17). GrubHub has shared its order data instance for on-demand meal delivery by transforming the geographic coordinates of customers and restaurants into UTM (Universal Transverse Mercator) coordinates.[3] We can leverage this dataset to estimate the delivery distance in real world. Because the simulation is conducted during the evening peak hours, we extract the data within the time period from 05:00 pm to 09:00 pm.

[3] https://github.com/grubhub/mdrplib.

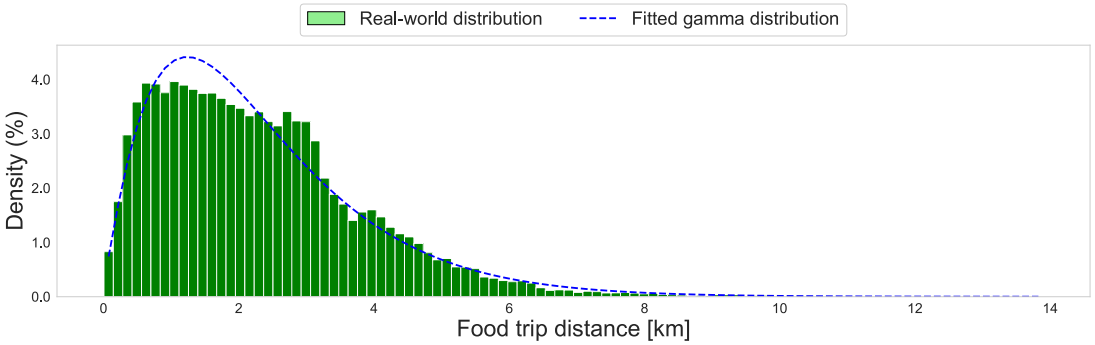


Fig. 17. Histogram and fitted distribution of food trip distances.

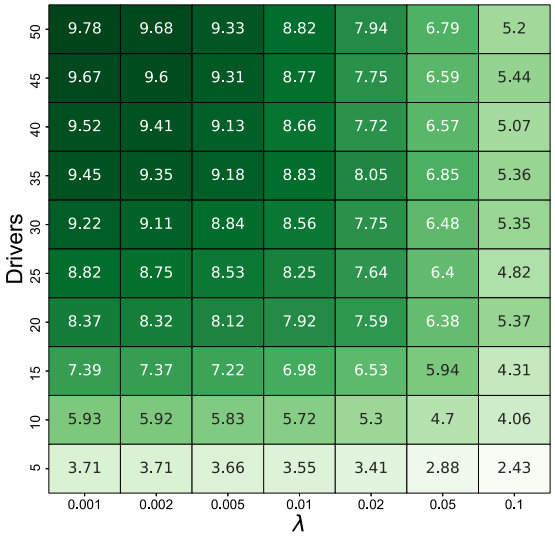


Fig. 18. Average objective values under different value of λ.

The data are cleaned by removing outlier orders with a distance over the 99% range of the dataset. Fig. 17 shows the distribution of the delivery distance and the fitted gamma distribution. The estimated shape and scale parameters of the fitted gamma distribution are approximately 2.18 and 1.07, respectively. We observe that over 99% of the delivery distances are shorter than 7 [km] with a very spread tail in the distribution.

Using a sampled order distance, the destination coordinate of each order is randomly selected from the points on the circle centred at the restaurant location with a radius of the sampled delivery distance. Finally, the fare of each order is computed by the rate from Uber Eats[4]:

$$\text{food order fare} = \overbrace{\$2.5}^{\text{pick-up fee}} + \overbrace{\$3.0}^{\text{drop-off fee}} + \overbrace{\$0.4}^{\text{distance unit fee}} \cdot \text{food trip distance [km]} \tag{57}$$

Appendix C. Parameter tuning

Algorithm 3 contains parameter λ, the value of which must be set before the algorithm is executed. To set the value of λ, we performed several experimental executions of Algorithm 3 using the data from Section 5.1. We consider six instances with ten passenger trip orders and ten different numbers of drivers (i.e., 5, 10, 15, 20, 25, 30, 35, 40, 45, 50). Order fares, pick-up times, and values of destination are sampled from the dataset. Driver–order utilities and *decline* utilities are generated based on the assumed preference parameters in the **Assortment** phase. Consequently, Algorithm 3 is applied 10 times to each of the ten instances for λ values taken from $\{0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1\}$.

[4] https://blog.completepayroll.com/fees-earnings-and-the-financial-side-of-ubereats.

Fig. 18 provides the average objective (i.e., the number of expected matches) obtained over the ten executions of each configuration. We can observe that the objective values are compromised with larger λ. Hence, The value of λ is set to 0.001, which indicated the best performance for these instances.

Appendix D. Cluster-based bundling algorithm

Let $\hat{O}^k$ denotes the orders from restaurant group k. Then we can proceed with bundling the orders within the same restaurant group (hexagon) by Algorithm 4. Algorithm 4 presents a distance-based clustering algorithm that incorporates a maximum cluster size (i.e., driver capacity) constraint. The objective of the algorithm is to partition meal orders in $\hat{O}^k$ into clusters based on geographical proximity while ensuring that no cluster exceeds a specified maximum size (i.e., driver capacity). Algorithm 4 starts by collecting the coordinates (latitude and longitude) of the destination coordinates L^k for orders in $\hat{O}^k$. The algorithm takes as input a radius ρ that defines the distance within which points are considered neighbours, and a maximum cluster size, MaxPts, that restricts the number of points allowed in each cluster. Algorithm 4 returns a list of cluster labels that indicate the assigned cluster for each order in $\hat{O}^k$.

Algorithm 4 Distance-Based Clustering with Maximum Cluster Size

Input: Destination coordinates L^k of $\hat{O}^k$, Radius limit (ρ), Maximum number of points (MaxPts)
Output: Optimal cluster label

```
Initialize cluster labels of L^k as [−1, −1, ..., −1]
Initialize cluster_index ← 0
for i ← 1 to length(L^k) do
    if labels[i] = −1 then
        Find all points within radius ρ around L^k[i]
        neighbors ← indices of points within radius ρ of L^k[i]
        if length(neighbors) > MaxPts then
            Split large cluster into subclusters
            subclusters ← Split large cluster using another clustering algorithm
            for each subcluster in subclusters do
                Assign points in subcluster to new cluster cluster_index
                cluster_index ← cluster_index + 1
            end for
        else
            Assign all points in the neighborhood to the same cluster cluster_index
            cluster_index ← cluster_index + 1
        end if
    end if
end for
return labels
```

Appendix E. Adaptive γ^*-greedy algorithm

We pre-calculate the values of γ^* given range of m and n up to a maximum of 200, and store these values in a look-up table. This tables allow for the adaptation of γ^* in Algorithm 5 to accommodate various combinations of m and n. We denote the value of γ^* for a given m and n in the table as $\gamma^*(m,n)$. We refer to this modified algorithm as the 'Adaptive γ^*-greedy' algorithm to distinguish it from the original version. The numerical results evaluated by the adaptive γ^*-greedy are shown in Table 4.

Note that γ^* is derived from the homogeneous case but is utilized to address the heterogeneous case in the numerical experiments. As a result, γ^* inherently carries a bias in both the γ^*-greedy and the adaptive γ^*-greedy. It is observed that the adaptive γ^*-greedy algorithm exhibits a slightly worse performance compared to the original γ^*-greedy algorithm in terms of both the number of matches and matching time. This discrepancy is attributed to the values of γ^*. It can be observed that the values of γ^* of the adaptive γ^*-greedy are significantly higher than $e-1$. This suggests that the adaptive γ^*-greedy algorithm is more inclined to favour the **Global menu** when determining the dispatch menus. In support of this, Fig. 19 depicts the menu sizes associated with the γ^*-greedy and adaptive γ^*-greedy algorithms. It is observed that the average menu size of the adaptive γ^*-greedy algorithm exceeds that of the γ^*-greedy algorithm. In contrast, the γ^*-greedy algorithm imposes a stricter condition (i.e., $\gamma^* = e-1$) to mitigate the overuse of the **Global menu**.

Algorithm 5 Adaptive γ^*-greedy algorithm

Input: O, D
Output: X
$X = \emptyset$, $O' = O$, $D' = D$
$(d^*, o^*) = \text{argmax}_{(d,o)}\{\gamma_{d,o}\}$
$m = |O'|$, $n = |D'|$
while $\gamma_{d^*,o^*} > \gamma^*(m,n)$ **do**
  $X \leftarrow X \cup \{x_{d^*,o^*}\}$
  $O' \leftarrow O' \backslash \{o^*\}$
  $D' \leftarrow D' \backslash \{d^*\}$
  $(d^*, o^*) = \text{argmax}_{(d,o)}\{\gamma_{d,o}\}$
  $m = |O'|$, $n = |D'|$
end while
for $o \in O'$ **do**
  for $d \in D$ **do**
    $X \leftarrow X \cup \{x_{d,o}\}$
  end for
end for

Table 4
The results of different assortment methods during 05:30 pm–09:00 pm (averaged over 10 test days). The numbers in the parentheses are the average response rate and the average cancellation rate of the platform. Note that the sum of the response and cancellation rates might be less than 100 percent. This is because some orders may still wait in the system at 09:00 pm.

Methods	Avg. match	Avg. cancellation	Avg. match time [s]	Avg. pick-up time [s]	Avg. occupied rate	Avg. leaving vehicles	Daily driver revenue		Avg. CPU time [s]
							Mean [USD]	Std [USD]	
Global menu	4081.0 (65.8%)	2113.0 (34.1%)	47.8	206.8	61.2%	31.6	129.7	41.3	0.0
Local menu	4194.0 (67.6%)	1973.0 (31.8%)	50.8	190.3	66.5%	71.3	130.8	51.9	0.0
One-to-one MTPT	4429.0 (71.4%)	1716.0 (27.7%)	57.4	93.5	67.6%	126.3	120.5	72.0	0.1
One-to-one MTSP	4453.0 (71.8%)	1724.0 (27.8%)	38.2	197.6	66.4%	55.8	136.8	47.5	0.1
SG for disjoint menu	4506.0 (72.6%)	1696.0 (27.3%)	35.7	277.7	61.0%	15.1	139.5	18.5	0.9
LS for joint menu	4693.0 (75.7%)	1510.0 (24.3%)	28.3	253.9	62.6%	15.8	143.4	20.7	7.8
γ^*-greedy	4542.0 (73.2%)	1660.0 (26.8%)	34.2	265.0	61.2%	16.2	140.0	17.9	0.1
Adaptive γ^*-greedy	4474.0 (72.1%)	1729.0 (27.9%)	37.1	228.2	61.2%	21.5	139.7	22.3	0.1

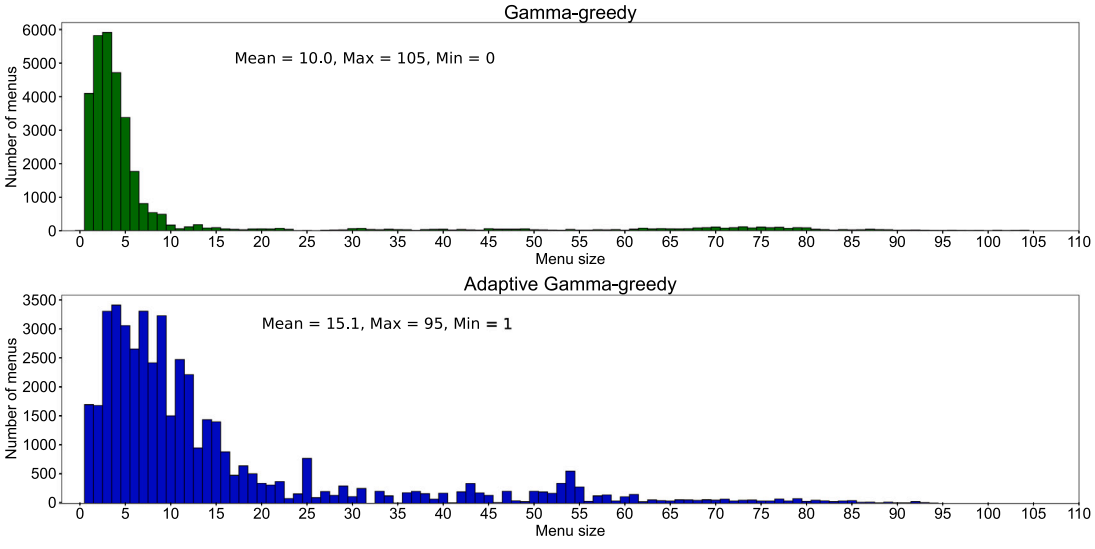


Fig. 19. Distributions of menu sizes of the γ^*-greedy and the adaptive γ^*-greedy algorithms.

References

Agatz, N., Erera, A.L., Savelsbergh, M.W., Wang, X., 2011. Dynamic ride-sharing: A simulation study in metro Atlanta. Procedia-Soc. Behav. Sci. 17, 532–550.

Agatz, N., Erera, A., Savelsbergh, M., Wang, X., 2012. Optimization for dynamic ride-sharing: A review. European J. Oper. Res. 223 (2), 295–303.

Alisoltani, N., Ameli, M., Zargayouna, M., Leclercq, L., 2022. Space-time clustering-based method to optimize shareability in real-time ride-sharing. PLoS One 17 (1), e0262499.

Alisoltani, N., Leclercq, L., Zargayouna, M., 2021. Can dynamic ride-sharing reduce traffic congestion? Transp. Res. B 145, 212–246.

Ashkrof, P., Correia, G.H.d., Cats, O., van Arem, B., 2021. Ride acceptance behaviour of ride-sourcing drivers. arXiv preprint arXiv:2107.07864.

Ashkrof, P., de Almeida Correia, G.H., Cats, O., van Arem, B., 2020. Understanding ride-sourcing drivers' behaviour and preferences: Insights from focus groups analysis. Res. Transp. Bus. Manage. 37, 100516.

Ashlagi, I., Krishnaswamy, A.K., Makhijani, R., Saban, D., Shiragur, K., 2022. Assortment planning for two-sided sequential matching markets. Oper. Res..

Association, A.A., et al., 2012. Your Driving Costs: How Much are You Really Paying to Drive. American Automobile Association, Miami, FL, USA.

Ausseil, R., Pazour, J.A., Ulmer, M.W., 2022. Supplier menus for dynamic matching in peer-to-peer transportation platforms. Transp. Sci..

Baldacci, R., Maniezzo, V., Mingozzi, A., 2004. An exact method for the car pooling problem based on lagrangean column generation. Oper. Res. 52 (3), 422–439.

Beojone, C.V., Geroliminis, N., 2021. On the inefficiency of ride-sourcing services towards urban congestion. Transp. Res. C 124, 102890.

Bogunovic, I., Zhao, J., Cevher, V., 2018. Robust maximization of non-submodular objectives. In: International Conference on Artificial Intelligence and Statistics. PMLR, pp. 890–899.

Chen, L., Valadkhani, A.H., Ramezani, M., 2021. Decentralised cooperative cruising of autonomous ride-sourcing fleets. Transp. Res. C 131, 103336.

Conforti, M., Cornuéjols, G., 1984. Submodular set functions, matroids and the greedy algorithm: tight worst-case bounds and some generalizations of the Rado-Edmonds theorem. Discrete Appl. Math. 7 (3), 251–274.

Cook, J., 2015. Uber's internal charts show how its driver-rating system actually works. Bus. Insider 12, 2015.

Dandl, F., Engelhardt, R., Hyland, M., Tilg, G., Bogenberger, K., Mahmassani, H.S., 2021. Regulating mobility-on-demand services: Tri-level model and bayesian optimization solution approach. Transp. Res. C 125, 103075.

Davis, J.M., Gallego, G., Topaloglu, H., 2014. Assortment optimization under variants of the nested logit model. Oper. Res. 62 (2), 250–273.

de Ruijter, A., Cats, O., Kucharski, R., van Lint, H., 2022. Evolution of labour supply in ridesourcing. Transportmetrica B: Transp. Dyn. 1–28.

Dickerson, J.P., Sankararaman, K.A., Srinivasan, A., Xu, P., 2017. Allocation problems in ride-sharing platforms: Online matching with offline reusable resources. arXiv preprint arXiv:1711.08345.

Einav, L., Farronato, C., Levin, J., 2016. Peer-to-peer markets. Annu. Rev. Econ. 8, 615–635.

Feige, U., Mirrokni, V.S., Vondrák, J., 2011. Maximizing non-monotone submodular functions. SIAM J. Comput. 40 (4), 1133–1153.

Fielbaum, A., Kucharski, R., Cats, O., Alonso-Mora, J., 2022. How to split the costs and charge the travellers sharing a ride? aligning system's optimum with users' equilibrium. European J. Oper. Res. 301 (3), 956–973.

Fielbaum, A., Tirachini, A., 2021. The sharing economy and the job market: the case of ride-hailing drivers in Chile. Transportation 48 (5), 2235–2261.

Fisher, M.L., Nemhauser, G.L., Wolsey, L.A., 1978. An analysis of approximations for maximizing submodular set functions—II. In: Polyhedral Combinatorics. Springer, pp. 73–87.

Fradkin, A., 2017. Search, matching, and the role of digital marketplace design in enabling trade: Evidence from airbnb. In: Matching, and the Role of Digital Marketplace Design in Enabling Trade: Evidence from Airbnb (March 21, 2017).

Group, E.D.R., 2018. Uber's economic impacts across the United States. URL https://drive.google.com/file/d/1P6HMbPc8T91Y8NlYyFGv8NQS9g4ckAq9/view.

Hamedmoghadam, H., Ramezani, M., Saberi, M., 2019. Revealing latent characteristics of mobility networks with coarse-graining. Sci. Rep. 9 (1), 1–10.

Hinxman, A., 1980. The trim-loss and assortment problems: A survey. European J. Oper. Res. 5 (1), 8–18.

Hong, S.J., Bauer, J.M., Lee, K., Granados, N.F., 2020. Drivers of supplier participation in ride-hailing platforms. J. Manage. Inf. Syst. 37 (3), 602–630.

Horner, H., Pazour, J., Mitchell, J.E., 2021. Optimizing driver menus under stochastic selection behavior for ridesharing and crowdsourced delivery. Transp. Res. E 153, 102419.

Isobel Asher Hamilton, M.H., 2019. PHOTOS: Uber drivers across the world are striking about pay, conditions, and the firm's 'orgy of greed'. URL https://www.businessinsider.nl/photos-uber-drivers-go-on-strike-around-the-world-2019-5?international=true&r=US.

Jiao, G., Ramezani, M., 2022. Incentivizing shared rides in e-hailing markets: Dynamic discounting. Transp. Res. C (ISSN: 0968-090X) 144, 103879. http://dx.doi.org/10.1016/j.trc.2022.103879, URL https://www.sciencedirect.com/science/article/pii/S0968090X22002923.

Jiao, G., Ramezani, M., 2024. A real-time cooperation mechanism in duopoly e-hailing markets. Transp. Res. C 162, 104598.

Lyu, G., Cheung, W.C., Teo, C.-P., Wang, H., 2019. Multi-objective online ride-matching. Available at SSRN 3356823.

Mancini, S., Gansterer, M., 2022. Bundle generation for last-mile delivery with occasional drivers. Omega 108, 102582.

Masoud, N., Jayakrishnan, R., 2017a. A decomposition algorithm to solve the multi-hop peer-to-peer ride-matching problem. Transp. Res. B 99, 1–29.

Masoud, N., Jayakrishnan, R., 2017b. A real-time algorithm to solve the peer-to-peer ride-matching problem in a flexible ridesharing system. Transp. Res. B 106, 218–236.

Mofidi, S.S., Pazour, J.A., 2019. When is it beneficial to provide freelance suppliers with choice? A hierarchical approach for peer-to-peer logistics platforms. Transp. Res. B 126, 1–23.

Munkres, J., 1957. Algorithms for the assignment and transportation problems. J. Soc. Ind. Appl. Math. 5 (1), 32–38.

Nemhauser, G.L., Wolsey, L.A., Fisher, M.L., 1978. An analysis of approximations for maximizing submodular set functions—I. Math. Program. 14 (1), 265–294.

Nourinejad, M., Ramezani, M., 2020. Ride-sourcing modeling and pricing in non-equilibrium two-sided markets. Transp. Res. B 132, 340–357.

Özkan, E., Ward, A.R., 2020. Dynamic matching for real-time ride sharing. Stoch. Syst. 10 (1), 29–70.

Pelzer, D., Xiao, J., Zehe, D., Lees, M.H., Knoll, A.C., Aydt, H., 2015. A partition-based match making algorithm for dynamic ridesharing. IEEE Trans. Intell. Transp. Syst. 16 (5), 2587–2598.

Pentico, D.W., 1974. The assortment problem with probabilistic demands. Manage. Sci. 21 (3), 286–290.

Qin, X., Yang, H., Wu, Y., Zhu, H., 2021. Multi-party ride-matching problem in the ride-hailing market with bundled option services. Transp. Res. C 131, 103287.

Ramezani, M., Nourinejad, M., 2018. Dynamic modeling and control of taxi services in large-scale urban networks: A macroscopic approach. Transp. Res. C 94, 203–219.

Ramezani, M., Valadkhani, A.H., 2023. Dynamic ride-sourcing systems for city-scale networks-Part I: Matching design and model formulation and validation. Transp. Res. C 152, 104158.

Ramezani, M., Yang, Y., Elmasry, J., Tang, P., 2022. An empirical study on characteristics of supply in e-hailing markets: a clustering approach. Transp. Lett. 1–14.

Research, Markets, 2022. Global online food delivery & takeaway market research report (2021 to 2026) - by food price range, food type, product type, distribution channel, application and region. URL https://www.prnewswire.com/news-releases/global-online-food-delivery--takeaway-market-research-report-2021-to-2026---by-food-price-range-food-type-product-type-distribution-channel-application-and-region-301499182.html.

Rosenblat, A., Stark, L., 2016. Algorithmic labor and information asymmetries: A case study of Uber's drivers. Int. J. Commun. 10, 27.

Sampaio, A., Savelsbergh, M., Veelenturf, L., Van Woensel, T., 2019. Crowd-based city logistics. In: Sustainable Transportation and Smart Logistics. Elsevier, pp. 381–400.

Smart, R., Rowe, B., Hawken, A., et al., 2015. Faster and cheaper: How ride-sourcing fills a gap in low-income los angeles neighborhoods.

Sui, Y., Zhang, H., Song, X., Shao, F., Yu, X., Shibasaki, R., Sun, R., Yuan, M., Wang, C., Li, S., et al., 2019. GPS data in urban online ride-hailing: A comparative analysis on fuel consumption and emissions. J. Clean. Prod. 227, 495–505.
Sungur, I., Ren, Y., Ordóñez, F., Dessouky, M., Zhong, H., 2010. A model and algorithm for the courier delivery problem with uncertainty. Transp. Sci. 44 (2), 193–205.
Tafreshian, A., Masoud, N., 2020. Trip-based graph partitioning in dynamic ridesharing. Transp. Res. C 114, 532–553.
Valadkhani, A.H., Ramezani, M., 2023. Dynamic ride-sourcing systems for city-scale networks, Part II: Proactive vehicle repositioning. Transp. Res. C 152, 104159.
Vondrák, J., 2007. Submodularity in combinatorial optimization.
Wang, X., Liu, W., Yang, H., Wang, D., Ye, J., 2019. Customer behavioural modelling of order cancellation in coupled ride-sourcing and taxi markets. Transp. Res. Procedia 38, 853–873.
Wang, H., Yang, H., 2019. Ridesourcing systems: A framework and review. Transp. Res. B 129, 122–155.
Wang, W., Zheng, J., Ren, F., Kannan, K.N., 2021. Estimating the value of destination disclosure: A dynamic structural model in a transportation network. Available at SSRN 3857243.
Xu, Z., A.M.C. Vignon, D., Yin, Y., Ye, J., 2020a. An empirical study of the labor supply of ride-sourcing drivers. Transp. Lett. 1–4.
Xu, Z., Li, Z., Guan, Q., Zhang, D., Li, Q., Nan, J., Liu, C., Bian, W., Ye, J., 2018a. Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 905–913.
Xu, K., Sun, L., Liu, J., Wang, H., 2018b. An empirical investigation of taxi driver response behavior to ride-hailing requests: A spatio-temporal perspective. PLoS One 13 (6), e0198605.
Xu, Z., Yin, Y., Ye, J., 2020b. On the supply curve of ride-hailing systems. Transp. Res. B 132, 29–43.
Yan, C., Zhu, H., Korolko, N., Woodard, D., 2020. Dynamic pricing and matching in ride-hailing platforms. Naval Res. Logist. 67 (8), 705–724.
Yang, H., Qin, X., Ke, J., Ye, J., 2020a. Optimizing matching time interval and matching radius in on-demand ride-sourcing markets. Transp. Res. B 131, 84–105.
Yang, Y., Ramezani, M., 2022. A learning method for real-time repositioning in E-Hailing Services. IEEE Trans. Intell. Transp. Syst. 1–11. http://dx.doi.org/10.1109/TITS.2022.3219381.
Yang, Y., Tian, Q., Wang, Y., 2020b. Who is more likely to get a ride and where is easier to be picked up in ride-sharing mode? J. Manage. Sci. Eng..
Yang, R., Xu, D., Du, D., Xu, Y., Yan, X., 2019b. Maximization of constrained non-submodular functions. In: International Computing and Combinatorics Conference. Springer, pp. 615–626.
Yang, L., Yu, X., Cao, J., Li, W., Wang, Y., Szczecinski, M., 2019a. A novel demand dispatching model for autonomous on-demand services. IEEE Trans. Serv. Comput..
Zhang, R., Masoud, N., 2021. A distributed algorithm for operating large-scale ridesourcing systems. Transp. Res. E 156, 102487.
Zhang, K., Nie, Y.M., 2021. Inter-platform competition in a regulated ride-hail market with pooling. Transp. Res. E 151, 102327.