Transportation Science

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Efficient Algorithms for Stochastic Ride-Pooling Assignment with Mixed Fleets

Qi Luo, Viswanath Nagarajan, Alexander Sundt, Yafeng Yin, John Vincent, Mehrdad Shahabi

To cite this article:
Qi Luo, Viswanath Nagarajan, Alexander Sundt, Yafeng Yin, John Vincent, Mehrdad Shahabi (2023) Efficient Algorithms for Stochastic Ride-Pooling Assignment with Mixed Fleets. Transportation Science 57(4):908-936. https://doi.org/10.1287/trsc.2021.0349

https://pubsonline.informs.org/journal/trsc

TRANSPORTATION SCIENCE
Vol. 57, No. 4, July–August 2023, pp. 908–936
ISSN 0041-1655 (print), ISSN 1526-5447 (online)

Efficient Algorithms for Stochastic Ride-Pooling Assignment with Mixed Fleets

Qi Luo,[a,*] Viswanath Nagarajan,[b] Alexander Sundt,[c] Yafeng Yin,[b,c] John Vincent,[d] Mehrdad Shahabi[d]

[a] Department of Industrial Engineering, Clemson University, Clemson, South Carolina 29634; [b] Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, Michigan 48109; [c] Department of Civil and Environmental Engineering, University of Michigan, Ann Arbor, Michigan 48109; [d] Ford Motor Company, Dearborn, Michigan 48120

*Corresponding author

Contact: qluo2@clemson.edu, https://orcid.org/0000-0002-4103-7112 (QL); viswa@umich.edu (VN); asundt@umich.edu (AS); yafeng@umich.edu, https://orcid.org/0000-0003-3117-5463 (YY); jvince60@ford.com (JV); mehrdad.shahabi@gmail.com (MS)

Received: August 12, 2021
Revised: March 31, 2022; November 29, 2022; April 23, 2023
Accepted: May 15, 2023
Published Online in Articles in Advance: June 15, 2023

https://doi.org/10.1287/trsc.2021.0349

Abstract. Ride-pooling, which accommodates multiple passenger requests in a single trip, has the potential to substantially enhance the throughput of mobility-on-demand (MoD) systems. This paper investigates MoD systems that operate mixed fleets composed of "basic supply" and "augmented supply" vehicles. When the basic supply is insufficient to satisfy demand, augmented supply vehicles can be repositioned to serve rides at a higher operational cost. We formulate the joint vehicle repositioning and ride-pooling assignment problem as a two-stage stochastic integer program, where repositioning augmented supply vehicles precedes the realization of ride requests. Sequential ride-pooling assignments aim to maximize total utility or profit on a shareability graph: a hypergraph representing the matching compatibility between available vehicles and pending requests. Two approximation algorithms for midcapacity and high-capacity vehicles are proposed in this paper; the respective approximation ratios are $1/p^2$ and $(e-1)/(2e+o(1))p\ln p$, where p is the maximum vehicle capacity plus one. Our study evaluates the performance of these approximation algorithms using an MoD simulator, demonstrating that these algorithms can parallelize computations and achieve solutions with small optimality gaps (typically within 1%). These efficient algorithms pave the way for various multimodal and multiclass MoD applications.

History: This paper has been accepted for the *Transportation Science* Special Issue on Emerging Topics in Transportation Science and Logistics.
Funding: This work was supported by the National Science Foundation [Grants CCF-2006778 and FW-HTF-P 2222806], the Ford Motor Company, and the Division of Civil, Mechanical, and Manufacturing Innovation [Grants CMMI-1854684, CMMI-1904575, and CMMI-1940766].
Supplemental Material: The online appendix is available at https://doi.org/10.1287/trsc.2021.0349.

Keywords: ride-pooling assignment problem • approximation algorithm • mixed autonomy traffic

1. Introduction

Ride-pooling assignment aims to dynamically determine the efficient dispatching of vehicles to handle multiple ride requests in a single ride in mobility-on-demand (MoD) systems. It generalizes various fleet management problems in applications ranging from ride-hailing (Santi et al. 2014) to microtransit (Li, Luo, and Hampshire 2021) and shared autonomous vehicles (Lokhandwala and Cai 2018). Efficient ride-pooling assignment algorithms can enhance the profitability of MoD services and increase the system throughput, that is, the number of completed customer trips per unit of time (Ke et al. 2021). Although consumers may experience trip delays due to detours, they are compensated by splitting the fare with coriders. More importantly, ride-pooling can decrease dead-heading trips that contribute to excessive energy use and greenhouse gas emissions of MoD platforms (Markov et al. 2021).

One of the MoD platform's central tasks is to achieve a dynamic balance between supply (available vehicles) and demand (pending ride requests). However, this balance is often unattainable due to supply shortages, such as a lack of freelance drivers during peak hours (Guda and Subramanian 2019), the inefficacy of empty-car cruising and searching for customers (Braverman et al. 2019), and drivers' perception errors regarding the supply-demand imbalance (Dong et al. 2021). Contrariwise, the heterogeneity of travel demand and driver types, as well as advancements in vehicle automation, have introduced the notion of "mixed fleet" into MoD platforms, which is illustrated by the following examples:

Example 1. Transportation network companies (TNCs) such as Uber and Didi Chuxing cater to diverse market segments by offering various service options. UberX is a standard service operated by freelancers, whereas Uber Black and Didi Chauffeur are premium services

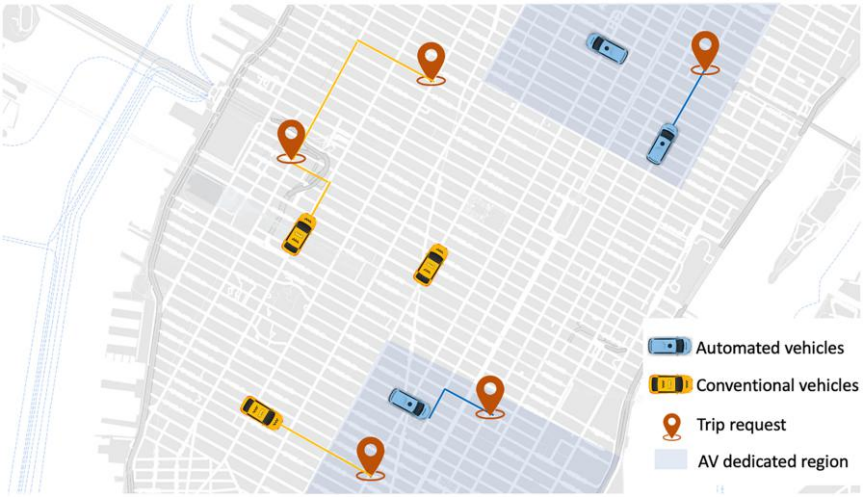
driven by professional drivers. Typically, the platform pairs users with their requested service class. However, when the standard class is in short supply, it may be advantageous for the platform to reposition premium vehicles to high-demand areas to fulfill standard-class ride requests and minimize cancellations.

Example 2. A mixed-autonomy platform operates both fully automated vehicles (AVs) owned by the platform and conventional vehicles (CVs) driven by human freelancers to provide on-demand transit services (Figure 1). When selecting the type of vehicle to dispatch, the operator must consider that (1) customers may prefer to be served by an AV or a CV, depending on their level of trust in automation (Lavieri and Bhat 2019), and (2) the accessible areas and operational costs of AVs and CVs for transporting customers may differ (Chen et al. 2017a, Shladover 2018).

Although these examples have distinct contexts, they can be generalized as the following *stochastic ridepooling assignment with mixed fleets* (SRAMF) problem. The mixed fleets consist of "basis supply" vehicles and "augmented supply" vehicles. Basis supply refers to vehicles operated by freelance drivers who use self-interested strategies when searching for customers, serve most customers in a decentralized manner, and presumably produce friction in balancing supply and demand (Dong et al. 2023). Augmented supply refers to vehicles (such as AVs) that follow the platform's centralized repositioning policies. Because of the different characteristics of supply sources, the platform faces a tradeoff between cost and control when matching ride requests with available vehicles. For a given level of demand, assigning nearby basis supply vehicles will incur lower operational costs than assigning augmented set vehicles. For example, the platform must pay salaries to full-time drivers in the augmented supply in Example 1 and costly maintenance costs for AVs in Example 2, which will be incorporated into the cost of serving each ride request. On the other hand, the platform may only have the authority to proactively reposition and reassign augmented supply vehicles to complement unsatisfied demand. As such, the platform's decision involves whether and where to reposition augmented supply vehicles, which primarily depends on the consequent assignment between available basis and augmented supply vehicles with realized ride requests.

Two unique operational challenges arise due to the diversification of vehicle fleets on MoD platforms. First, operating MoD with mixed fleets face inherent *uncertainties* in the sequential vehicle repositioning and ride-pooling assignment processes as follows. In the first-stage vehicle repositioning decisions, the platform forecasts future demand and repositions selected premium service vehicles (Example 1) or AVs (Example 2) to specific locations to accommodate unmet demands for the basis supply. In the second-stage ride-pooling assignment decisions, the platform assigns realized ride requests to available vehicles, including basis and augmented supply, to maximize the total value of assignments. The uncertainties between the vehicle repositioning and assignment stages can be categorized as supply-based or demand-based factors. Supply-based uncertainty concerns whether basis-supply drivers stay active in future periods. Demand-based uncertainty includes the origins and destinations of upcoming ride requests, the number of passengers per order, customers' unknown preferred vehicle type, and their value of time. Because falsely repositioned

Figure 1. (Color online) Ride-Pooling with AVs and CVs



Note. The first-stage decision involves repositioning AVs in dedicated regions; the second-stage decision is to solve a general assignment problem (GAP).

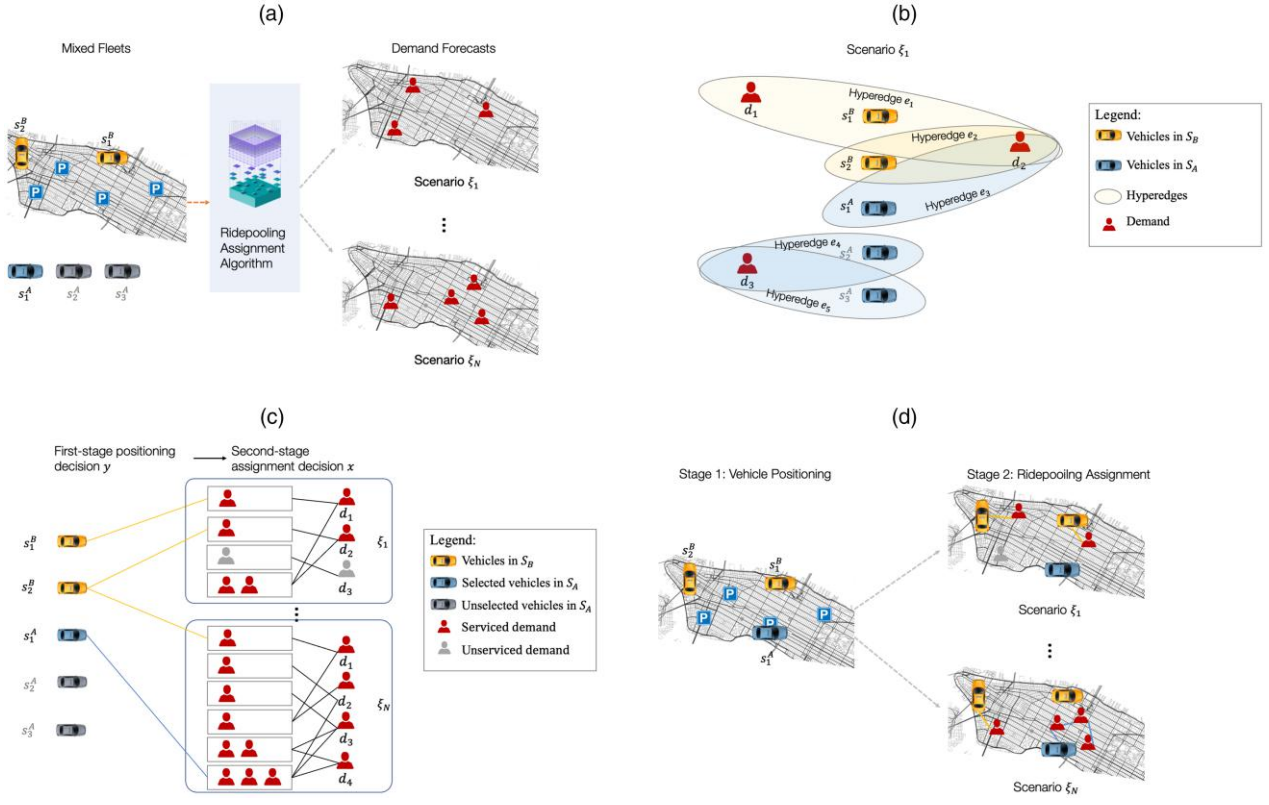
vehicles will result in a supply-demand mismatch in the future, joint repositioning and assignment will cause complicated tradeoffs in MoD operations.

Second, previous aggregate vehicle repositioning models are not implementable for vehicle-level operation in MoD systems. The SRAMF problem differs from the large body of mixed-fleet planning literature (Guo, Caros, and Zhao 2021; Karamanis et al. 2021) that used an aggregate matching model in region-to-region repositioning flow computations. Focusing on vehicle-level operations under uncertainty will cause significantly more computational burden than the aggregate setting. This *scalability* issue intensifies as the platform uses high-capacity augmented supply vehicles to compensate for their high operational costs, such as on-demand transit services (Hasan and Van Hentenryck 2021). With expanded vehicle capacity, the number of candidate pickup and dropoff routes can grow exponentially. These unique technical challenges of joint repositioning and assignment decisions motivate the development of effective and efficient SRAMF algorithms in this study.

This study expands the deterministic ride-pooling assignment of homogeneous vehicle fleets in Santi et al. (2014) and Alonso-Mora et al. (2017) to a stochastic setting in a nontrivial way. The scalable framework addresses the computational challenge of the second-stage problem in SRAMF by separating the vehicle routing and trip-to-vehicle assignment into two sequential steps based on the notion of "shareability graphs." Specifically, given ride requests and available vehicles in each time interval, Alonso-Mora et al. (2017) proposed a procedure that guaranteed anytime optimality; that is, the resulting ride-pooling assignments attain the same solutions as the integrated vehicle routing and trip assignment formulation (see Appendix B, Section B.1). The procedure is summarized as follows:

1. First, the algorithm constructs a shareability graph that represents the matchable relationship between all ride requests (demand) and available vehicles (supply) (Figure 2) and computes the value associated with each matching.

Figure 2. (Color online) SRAMF Procedure per Step



Notes. $S_B = \{s_1^B, s_2^B\}$ is the basis set (e.g., CVs) and $S_A = \{s_1^A, s_2^A\}$ is the augmented set (e.g., AVs). (a) Algorithm's input, including the current locations of S_A and S_B, and obtains demand forecast. (b) Shareability graph for each scenario, where each trip is a clique containing one vehicle and multiple matchable requests. (c) SRAMF problem by approximation algorithms, in which one or more ride requests are assigned to a selected vehicle in each scenario ξ. (d) Computed decisions and updates the system state.

2. Next, the algorithm maximizes the total matching value by solving a general assignment problem (GAP) on the shareability graph.

3. Finally, the shareability graph is updated by deleting occupied vehicles and assigned demand and substituting them with incoming requests and available vehicles.

The SRAMF problem can be formulated as a two-stage stochastic integer program. Incorporating repositioning decisions into the deterministic ride-pooling assignment is difficult due to the relationship between these consecutive steps. Because the vehicle repositioning decision must select augmented supply vehicles by repositioning them from the augmented supply set (a set of candidate locations) before the realization of demand, convoluted tradeoffs must be made between the first- and second-stage decisions. If the platform underestimates demand and selects fewer vehicles, it cannot meet all future requests. If the platform overestimates demand and selects more vehicles than needed, it must pay extra operational costs. In addition, because of various sources of uncertainties stated previously, the size of the shareability graph grows rapidly with the number of scenarios sampled. As a result, solving GAP in SRAMF using exact methods becomes inefficient or even infeasible.

1.1. Main Results and Contribution

The primary objective of this study is to develop approximation algorithms for solving large-scale SRAMF problems. We focus on the expected value maximization setting for several reasons. First, real-world concerns emphasize the need to enhance MoD systems' throughput and profitability (Ashlagi et al. 2018; Simonetto, Monteil, and Gambella 2019). Second, devising approximation algorithms for maximizing GAPs tends to be more challenging than minimizing GAPs (Fleischer et al. 2006). The objective function of SRAMF can incorporate various attributes, such as trip fares, pickup times, and ride-pooling preferences. The primary performance metric for the proposed algorithms is the *tightness* of approximation ratios, offering a provable performance guarantee in worst-case scenarios.

To summarize our work, let p denote the mixed fleets' largest vehicular capacity plus one. Our main results are as follows:

1. The SRAMF problem is proved to be NP-hard for any finite number of scenarios, and its objective lacks attractive *submodular* properties. These characteristics necessitate the development of new approximation algorithms to exploit the computational advantages of shareability graphs.

2. Our analysis provides provable worst-case performance guarantees as follows:

(a) For midcapacity vehicles, we develop a local-search linear-program-relaxation (LSLPR) algorithm, with an approximation ratio of $1/p^2$. Midcapacity vehicles carrying up to four passengers simultaneously are suitable for applications in Example 1.

(b) For high-capacity vehicles, we develop a max-min online (MMO) algorithm, with an approximation ratio of $(e-1)/(2e+o(1))p\ln p$. High-capacity vehicles carrying more than four requests are suitable for automated transit services in Example 2.

(c) These approximation ratios are close to the best possible bounds: no polynomial-time algorithm can achieve a ratio better than $O(\ln p/p)$ under standard complexity assumptions.

Our methods rely on a linear relaxation of the second-stage GAP and carefully bound the integrality gap of the relaxation in each scenario. Additionally, this analysis explains the sources of computational intractability of SRAMF and recognizes the significance of considering uncertainties per assignment.

This study contributes to the literature on MoD system operations as follows:

1. Propose a two-stage stochastic integer program for SRAMF and propose approximation algorithms with satisfactory performance guarantees. These easy-to-implement algorithms can facilitate fleet operations on MoD platforms and guarantee their performance in the face of uncertainties with provable bounds.

2. Derive a general estimator for marginal values of trip-to-vehicle matchings. The primary analytical barrier for the design of approximation algorithms for SRAMF is to evaluate the expected value of repositioning additional vehicles to serve future demand in a specific area. Our proof bounds this value and is of independent interest to relevant literature, for example, fleet sizing in MoD systems (Benjaafar et al. 2021).

3. Provide analytical solutions for fractional hypergraph matchings. Our analysis for the MMO algorithm derives a closed-form solution for the dual problem of fractional hypergraph matchings to accelerate enumerations. This closed-form solution can be transferred to other decomposition-based ride-pooling assignment methods.

We conducted comparative studies to illustrate the computational efficiency and optimality gaps of our developed algorithms using real-world taxicab trip data (TLC 2021). Numerical results showed our algorithms to be almost as competitive as mixed integer programming (MIP), indicating that the derived worst-case approximation ratios are conservative. This framework can incorporate various demand forecasts (Yang et al. 2020) and use state-dependent matching intervals (Qin et al. 2021). Moreover, our results extend to mixed fleets of more than two vehicle types.

1.2. Organization and General Notation

The remainder of the paper is organized as follows. We first review the related literature in Section 2. Section 3 formulates the SRAMF problem and shows its hardness. Section 4 proposes two approximation algorithms that achieve nearly tight approximation ratios. We test the

effectiveness of these approximation algorithms using real-world and simulated data in Section 5 and draw conclusions in Section 6.

The following notation is used throughout this work. The notation := stands for "defined as". For any integer n, we let $[n] := \{1, 2, \ldots, n\}$. We use $v(\cdot)$ as the actual value function and $\hat{v}(\cdot)$ as the approximate or estimated value function. P stands for the class of questions for which some algorithm can provide an answer in polynomial time, and NP stands for those with nondeterministic polynomial time algorithms. For any set S, $|S|$ is its cardinality. Given two sets A and B, $A + B$ or $A \cup B$ represents the union of A and B; $A - B$ or $A \backslash B$ represents modifying A by removing the elements belonging to B. $A \sim B$ represents that set A intersects with B, that is, $A \cap B \neq \emptyset$. i.i.d. stands for "independent and identically distributed." Other notation and acronyms used in this paper are summarized in Table A.1.

2. Literature Review

We refer to ride-pooling (also called ride-splitting/carsharing rides) in the broad context and focus on operations-level decisions. Solving the optimal ride-pooling assignment is challenging because the number of possible shared trips grows exponentially in the vehicle capacity and matching intervals. The following review covers the recent development of computational methods for ride-pooling applications with different objectives of maximizing the utilization of vehicles or reducing the negative externalities related to deadhead miles.

2.1. Decomposition and Approximate Dynamic Programming Approaches

Compared with the substantial body of literature for matching supply and demand without the ride-pooling option (Wang and Yang 2019), there are only a few attempts to solve the ride-pooling assignment problem at the vehicle level by combining heuristic and decomposition methods (Herminghaus 2019, Yu and Shen 2019, Sundt et al. 2021). Although these heuristics achieved satisfying performance in numerical experiments, they cannot balance computational efficiency and accuracy with theoretical guarantees. The trip planning for ride-pooling is more tractable with fixed travel patterns, such as providing services for daily commuting. Hasan, Van Hentenryck, and Legrain (2020) proposed a commute trip-sharing algorithm that maximized total shared rides for a set of commute trips satisfying various time-window, capacity, pairing, ride duration, and driver constraints.

Another stream of papers emphasized the importance of nonmyopic policies in MoD systems, as supply and demand dynamics are influenced by prior decisions. Unfortunately, because of the computational complexity, most nonmyopic ride-pooling assignment policies are restricted to aggregate models and compute optimal flows between regions. Shah, Lowalekar, and Varakantham (2020) developed an approximate dynamic programming method to learn from the integer program (IP)-based assignment and approximate the value function by neural networks. We refer readers to a comprehensive review Qin, Zhu, and Ye (2021) of reinforcement learning methods for ride-sharing assignments and other sequential decisions.

2.2. Deterministic Ride-Pooling Assignment for Shareability Graphs

To tackle those unprecedented computational challenges in MoD systems, Santi et al. (2014) quantified the tradeoff between social benefits and passenger discomfort from ride-pooling by introducing the concept of "shareability networks." They found that the total empty-car travel time was reduced by 40% in the offline setting (i.e., with ex post demand profiles) or 32% when demand is revealed en route. This work suffers a limitation in vehicle capacity as the matching-based algorithm can only handle up to three-passenger shared rides. Alonso-Mora et al. (2017) expanded the framework to up to 10 riders per vehicle. The high-capacity ride-pooling trip assignment is solved by decomposing the shareability graph into trip sets and vehicle sets and then solving the optimal assignments by a large-scale IP. As the vehicle capacities increase, the moderate size of the shared vehicle fleet (2,000 vehicles with capacities of four rides in their case studies) can serve most travel demands with short waiting times and trip delays. Simonetto, Monteil, and Gambella (2019) improved this approach's computational efficiency by formulating the master problem as a linear assignment problem. The resulting large-scale assignment on shareability networks is calculated in a distributed manner. However, despite the easy implementation of these methods, they lack theoretical performance guarantees.

2.3. Approximation Algorithms for Maximization GAP

Approximation algorithms can find near-optimal assignments with provable guarantees on the quality of returned solutions. Because the ride-pooling assignment problem is a variant of GAP (Öncan 2007), we list the significant results here. Shmoys and Tardos (1993) and Chekuri and Khanna (2005) obtained polynomial-time $\frac{1}{2}$-approximation algorithms. Fleischer et al. (2006) obtained an linear programming (LP)-rounding based $(1 - (1/e))$-approximation algorithm and a local-search based 1/2-approximation algorithm. Previous studies have explored GAP algorithms for both instant and batched dispatching settings. Instant dispatching assigns requests to available vehicles on arrival. Lowalekar,

Varakantham, and Jaillet (2020) developed approximation algorithms for online vehicle dispatch systems. Their setting with i.i.d. demand assumptions are markedly different from the current work. Batched dispatching uses GAP on a hypergraph to search for locally optimal assignments. Mori and Samaranayake (2021) developed $1/e$-approximation LP-rounding algorithms for the deterministic request-trip-vehicle assignment problem. In contrast, the current work considers a stochastic setting in which sequential repositioning and assignment decisions jointly determined the objective in SRAMF. As a batched dispatching algorithm, this stochastic formulation can be applied to arbitrary demand distributions.

2.4. Shared Mobility with Mixed Fleets

Mixed-fleet ride-sharing systems are emerging research topics in literature. The first stream of research is motivated by MoD platforms' transition to a blended workforce of permanent employees and freelance workers. Dong and Ibrahim (2020) investigated the staffing problem in which a ride-hailing platform determined the number of fore-hire drivers considering its impact on other flexible workers. Dong et al. (2021) justified the dual-source strategy for mitigating the demand uncertainty in ride-hailing systems and designed optimal contracts to coordinate the mixed workforce. Castro et al. (2020) modeled the ridesharing market as matching queues where drivers had different flexibility levels. They proposed a robust throughput-maximizing capacity reservation policy against the unknown driver engagement function.

The introduction of automation in MoD systems in the foreseeable future motivates a second stream of mixed-fleet research. Lokhandwala and Cai (2018) used agent-based simulations to evaluate the impact of heterogeneous preferences and revealed that the transition to a mixed fleet would reduce the total number of vehicles, focus on areas of dense demands, and lower the overall service levels in the suburban regions. Wei, Pedarsani, and Coogan (2020) studied the equilibrium of a mixed autonomy network in which AVs are fully controlled by the platform and CVs are operated by individual drivers. The optimal pricing for the mixed service is formulated as a convex program. Li, Chen, and Zhang (2022) proposed a traffic network equilibrium model with mixed autonomy based on two-player games and proved the existence of a speed policy that guarantees Pareto-efficient equilibria. Xie, Liu, and Chen (2023) developed an actor-critic learning approach for mixed-autonomy fleet management considering bounded rational drivers. In contrast, this work is one of the first attempts to develop algorithms for mixed-autonomy operations at the vehicle level.

3. Problem Description

3.1. Basic Setting

This section introduces the formulation of the SRAMF problem as a two-stage stochastic integer program and shows its NP-hardness. These technical challenges motivate the design of new approximation algorithms in the remainder of this work.

3.1.1. Preliminaries: Constructing a Shareability Graph of Mixed Fleets.

Ride-pooling assignment is conducted on a shareability graph, represented by a *hypergraph* $G = \{S, D, E\}$. The vertices of the hypergraph are $S \cup D$, where S denotes supply (available vehicles) and D denotes demand (ride requests). Each *hyperedge/clique* $e \in E$ consists of one vehicle and a subset of ride requests. In conventional assignments, each vehicle can serve only one ride request at a time, so G reduces to a bipartite graph. In the ride-pooling setting, each hyperedge $e \in E$ can contain any number of ride requests within the vehicle's capacity. Other constraints, such as the upper bounds for detour times, are considered when constructing the shareability graph (we refer readers to the discussion of shareability graphs in Appendix B). The platform continuously updates such a hypergraph following the procedure outlined in Section 1.1. Appendix B also describes a sequence of matching rules that can construct a hierarchical tree of matchable requests, significantly reducing the computational burden of dial-a-ride problems.

This generic model covers most MoD applications described in Section 1. The mixed-fleet supply contains a set S_A of locations to reposition augmented vehicles and a set of basis vehicles S_B. We assume that each augmented vehicle can reposition to any of the locations S_A and serve nearby ride requests covered by their incident hyperedges. Let $S = S_A \cup S_B$, $|S_A| = n_A$, and $|S_B| = n_B$. To keep notation simple, we will refer to the "locations to reposition augmented vehicles" S_A simply as the augmented supply/vehicles. We denote $p = 1 + \max_{i \in S_A \cup S_B} \{C_i\}$ where C_i is the capacity of vehicle i. Without loss of generality, we let the cost of using vehicles in S_B be zero and the cost of each vehicle in S_A be normalized to one. This will be extended to a more general setting of partition constraints in Section 4.3. The varying setup costs of S_A and S_B can be justified by the additional operations expenditure of repositioning centralized-controlled vehicles in the augmented set S_A, such as the annualized extra salary paid to full-time drivers in Example 1. Each hyperedge $e = \{i, J\}_{i \in S, J \subseteq D}$ corresponds to a potential trip where vehicle i serves ride requests in J.

The MoD platform will implement SRAMF algorithms using the online procedure outlined below. The platform first predicts available vehicles in S_B and ride requests D per batch and then constructs a shareability graph according to the procedure outlined in Appendix B, Section B.1. After calculating the value of each hyperedge v_e, the platform solves a two-stage stochastic integer program to determine the optimal centralized repositioning policy for vehicles in S_A. The platform then observes actual demand and vehicle locations and updates the shareability graph. The remainder of this section

formally defines the SRAMF problem and highlights its unique technical challenges.

3.1.2. Formulation of SRAMF. Before actual ride requests are sent, the platform chooses a subset $S_R \subset S_A$ of (at most) K locations to reposition vehicles from the augmented supply. After requests are revealed, the platform can assign ride requests only to vehicles in $S_R \cup S_B$ and collect instantaneous rewards (profits) from completing these trips; that is, reassignment is not allowed.

The sequential decisions for the SRAMF problem are as follows:

1. In the first stage, for each augmented vehicle $i \in S_A$, $y_i = 1$ denotes that an augmented vehicle is allocated to location i for future assignment and $y_i = 0$ denotes not selected. Let $S_R := \{i \in [n_A] : y_i = 1\} \subseteq S_A$ denote a set of selected augmented vehicles. All basis supply vehicles are included, as they impose no additional setup cost, and the available supply in the second stage is $S_R \cup S_B$. The first-stage decision space is $Y \in \{0,1\}^{n_A+n_B}$.

2. In the second stage, a scenario ξ reveals a set of actual ride requests $D(\xi)$ and their associated hyperedges $E(\xi)$. The scenario ξ is assumed to follow a random distribution $F(\xi)$ with support on Ξ, which incorporates a demand forecast model. Each hyperedge $e \in E(\xi)$ includes a vehicle i from either basis or augmented supply and a subset of requests $J \subset D(\xi)$. $\{w_j\}_{j\in J}$ denotes the numbers of passengers in each ride request j. The total number of passengers in a set of ride requests J must satisfy $\sum_{j\in J} w_j \le C_i$ where C_i is the capacity of vehicle i. The hyperedge value of e may include the following elements:

(a) The profit u_j gained from serving the ride request j.

(b) A trip $t = \{j_1, j_2, \cdots : j_k \in J\}$ represents a sequence of picking up ride requests in J. The associated travel cost $c(i, t)$ assumes that the vehicle i follows the shortest pick-up trip to minimize customers' waiting times.

(c) Each request j gains additional utility $\tilde{u}_{ij}$ if matched with their preferred vehicle type.

The hyperedge value for $e \in E(\xi)$ collected from a potential assignment is given by

$$v_e = \sum_{j\in J} u_j + \sum_{j\in J} \tilde{u}_{ij} - c(i,t) \ge 0. \tag{1}$$

The hyperedge value captures various sources of uncertainties between vehicle repositioning and trip assignment stages. u_j considers the uncertain number of ride requests and their origin and destination; w_j and the set J considers the unknown number of passengers in each ride request; $\tilde{u}_{ij}$ considers the customers' uncertain preference for vehicle types. Finally, because of fluctuating traffic conditions and different vehicle technology (e.g., CVs and AVs), $c(i, t)$ represents that pickup times are uncertain. However, in the second stage (after the scenario ξ is observed), all hyperedge values are known precisely. It is worth mentioning that the calculation of hyperedge values can be integrated with advanced value function approximation techniques. For example, Tang et al. (2019) calculated the associated hyperedge value as a reward signal derived from a reinforcement learning–based estimator.

3. The platform assigns ride requests to each available vehicle by determining $x_e \in \{0,1\}$ for all $e \in E(\xi)$. The second-stage assignment decision is equivalent to choosing a set of hyperedges in which every pair of hyperedges is disjoint. This condition guarantees that each vehicle and each ride request can be included no more than once in the final assignment per scenario. An assignment is only feasible between the chosen supply $S_R \cup S_B$ (denoted as $e \sim S_R \cup S_B$) and realized demand $D(\xi)$ in each scenario.

The optimal value of assignments in scenario ξ is calculated by $Q : Y \times \Xi \to \mathbb{R}$. Given a scenario, the second-stage decisions are trip assignments denoted by $x = \{x_e\}_{e\in E(\xi)}$. Our objective is to maximize the *expected total value*.

The SRAMF problem can be formulated as a two-stage stochastic integer program:

$$\underset{y}{\text{maximize}} \quad \mathbf{E}[Q(y,\xi)] \tag{2}$$

$$\text{s.t.} \sum_{i\in S_A} y_i \le K \qquad (\textit{budget}), \tag{2a}$$

$$y_i \in \{0,1\} \qquad \forall i \in S_A, \tag{2b}$$

$$y_i = 1 \qquad \forall i \in S_B, \tag{2c}$$

and the second-stage problem is given by

$$Q(y,\xi) = \underset{x}{\text{maximize}} \sum_{e\in E(\xi)} v_e x_e \tag{3}$$

$$\text{s.t.} \sum_{e\in E(\xi): j\in e} x_e \le 1 \quad \forall j \in D(\xi) \quad (\textit{assignment I}), \tag{3a}$$

$$\sum_{e\in E(\xi): i\in e} x_e \le y_i \quad \forall i \in S_B \cup S_A \quad (\textit{assignment II}), \tag{3b}$$

$$x_e \in \{0,1\} \qquad \forall e \in E(\xi). \tag{3c}$$

In the first-stage problem (2), K is the maximum number of locations for repositioning augmented supply vehicles. In the second-stage problem (3), Constraints (3a) and (3b) guarantee that each supply and demand is matched at most once and the vehicles selected in $S_R \cup S_B$ are matchable. In other words, unassigned vehicles and ride requests in the hypermatching x will either renege or postpone to the next batch. The second-stage GAP is a p-set packing problem with p representing the maximum size of hyperedges, which is known to be

NP-hard (Füredi, Kahn, and Seymour 1993; Chan and Lau 2012).

3.1.3. Roadmap for Proving SRAMF Approximation Algorithms.

Figure 3 provides an overview of the performance analysis of two proposed approximation algorithms and their approximation ratios, respectively. We start with reducing the objective of (2) to the sample-average estimate in Section 3.2. We then show the hardness of the SRAMF problem in Section 3.3. Because the GAP problem (3) is NP-hard, our approximation algorithms rely heavily on the "fractional assignment" technique that relaxes the integrality constraints in (3) as a polynomial-time solvable linear program. Two different approximation algorithms, LSLPR and MMO, are discussed in detail in Section 4.1 and Section 4.2, respectively.

3.2. Reduction to Sample-Average Estimate

The sample-average approximation (SAA) method is commonly used to solve two-stage stochastic integer programs. It draws N scenarios $\{\xi_\ell\}_{\ell=1}^N$ from a scenario-generating oracle (e.g., demand forecasting and vehicle simulation models) and approximates the expected objective function by a sample-average estimate $\mathbf{E}[Q(y,\xi)] \approx \sum_{\ell=1}^N Q(y,\xi_\ell)/N$.

To simplify the analysis of Problem (2), we reduce the objective function $\mathbf{E}(Q(y,\xi))$ to finite-sample proximity. The main analysis is conditional on N mutually disjoint sets of ride requests $D(\xi)$ and hyperedges $E(\xi)$. Because the second-stage assignment ensures unique matchings per scenario, we can make multiple disjoint copies when an identical ride request appears in multiple scenarios. The consistency and shrinking bias of the sample-average estimate are well studied in literature; hence, the proof of SAA is detailed in Appendix C, Section C.1, for completeness. Altogether, the optimal value of any approximation algorithm converges to $\mathbf{E}[Q(y,\xi)]$ as the number of scenarios $N \to \infty$.

This study's focus is therefore developing algorithms to solve the SRAMF problem in (2) with the sample-average estimate. As mentioned earlier, we will work with an LP relaxation of (3) as the original p-set packing problem is NP-hard. For any subset $S_R \subseteq S_A$ and scenario ξ, define $\hat{v}(S_R,\xi)$ to be the optimal value of the following LP:

$$\underset{x}{\text{maximize}} \sum_{e\in E(\xi)} v_e x_e \tag{4}$$

$$\text{s.t.} \sum_{e\in E(\xi): j\in e} x_e \le 1 \qquad \forall j \in D(\xi), \tag{4a}$$

$$\sum_{e\in E(\xi): i\in e} x_e \le 1 \qquad \forall i \in S_A \cup S_B, \tag{4b}$$

$$x_e = 0 \qquad \forall e \sim S_A \setminus S_R, \tag{4c}$$

$$x_e \ge 0 \qquad \forall e \in E(\xi). \tag{4d}$$

Solutions to the LP relaxation of (3) are called *fractional assignments*; $v(S_R,\xi)$ denotes the optimal value of exact solutions to (3), given a set of selected augmented supply S_R. Furthermore, we define two objective functions related to the sample average estimate:

- The objective value using the exact GAP in (3) for each scenario is given by

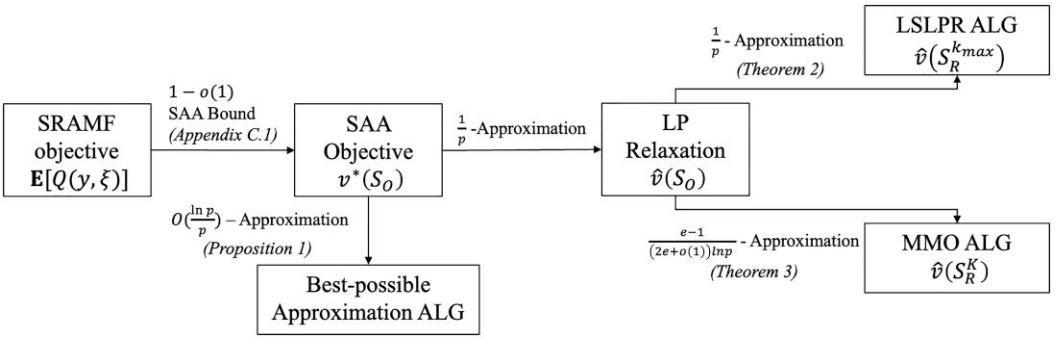
$$v^*(S_R) = \frac{1}{N}\sum_{\ell\in[N]} v(S_R,\xi_\ell). \tag{5}$$

- The objective value using the LP relaxation of (4) is given by

$$\hat{v}(S_R) = \frac{1}{N}\sum_{\ell\in[N]} \hat{v}(S_R,\xi_\ell). \tag{6}$$

Fractional assignments of the p-set packing problem enjoy the following properties. (1) The integrality gap between the exact solution and LP relaxation is at most p times (Arkin and Hassin 1998). (2) A greedy algorithm selecting hyperedges e in decreasing order of their values v_e while maintaining feasibility achieves a $1/p$-approximation to the LP value. We restate them in the following theorem.

Figure 3. Roadmap for the Performance Analysis on SRAMF Algorithms



Notes. The approximation ratios on arrows refer to the results in this paper. S_O is the optimal selection of vehicles, and S_R is the section of vehicles generated by approximation algorithms.

Theorem 1. *For any $S_R \subseteq S_A$, we have $v^*(S_R) \le \hat{v}(S_R) \le p \cdot v^*(S_R)$; furthermore, the greedy algorithm obtains a solution of value at least $1/p \cdot \hat{v}(S_R)$.*

These reductions narrow down the main task of bounding the approximation ratio of $\hat{v}(S_R)$. In particular, we will focus on the SRAMF problem with fractional assignments:

$$\max_{S_R \subseteq S_A : |S_R| \le K} \hat{v}(S_R). \tag{7}$$

If we obtain an α-approximation algorithm for (7), then combine it with Theorem 1, we would obtain an α/p-approximation algorithm for SRAMF (with integral assignments). Also, observe that the objective in (7) is monotone nondecreasing in the selected vehicle set S_R. Therefore, any maximal solution (including the optimal solution) selects exactly K vehicles in the repositioning decision. Before jumping into the design of approximation algorithms, the following section elaborates on some technical challenges.

3.3. Hardness and Properties of SRAMF

We show that solving SRAMF is computationally challenging due to the following reasons: (1) Proposition 1 shows that the second-stage assignment problem is NP-hard. Hence, computing the exact assignment for any S_R is costly. (2) Proposition 2 shows that $\hat{v}(S_R)$ is *not* submodular, preventing the use of efficient submodular maximization algorithms. These facts motivate the development of new approximation algorithms in Section 4 to exploit the specific structure of the SRAMF problem.

Proposition 1. *There is no algorithm for SRAMF (even with $n = 1$ scenario) with an approximation ratio better than $O(\ln p/p)$, unless $P = NP$.*

Proof for the Hardness of SRAMF. We reduce from the *p-dimensional matching* problem, defined as follows. There is a hypergraph H with vertices V partitioned into p parts $\{V_r\}_{r=1}^p$, and hyperedges E. Each hyperedge contains exactly one vertex from each part (so each hyperedge's size is p). The goal is to find a collection F of disjoint hyperedges that have maximum cardinality $|F|$.

Given any p-dimensional matching as above, we generate the following SRAMF instance. The augmented vehicles are $S_A = V_1$ and the basis vehicles are $S_B = \emptyset$. There is $n = 1$ scenario with ride requests $V_2 \cup \ldots V_p$ and hyperedges E (each of value of one). Each vehicle has a capacity of $p - 1$, and each ride request has one or more passengers. Each hyperedge contains precisely one vehicle, as required in SRAMF. The bound $K = |S_A|$ so the optimal first stage solution is clearly $S_R = S_A$ (select *all* locations for augmented vehicles). Now, the SRAMF problem instance reduces to its second-stage problem (3), which involves selecting a maximum cardinality subset of disjoint hyperedges. This is precisely the p-dimensional matching problem.

It follows that if there is any α-approximation algorithm for SRAMF with $n = 1$ scenario, then there is an α-approximation algorithm for p-dimensional matching. Finally, Hazan, Safra, and Schwartz (2006) proved that it is NP-hard to approximate p-dimensional matching better than an $O(\ln p/p)$ factor (unless $P = NP$). The proposition now follows. □

This intractability is the reason that we work with the *fractional* assignment problem (7). A natural approach for budgeted maximization problems such as (7) is to prove that the objective function is *submodular*, in which case one can directly use the $(1 - (1/e))$-approximation algorithm by (Nemhauser, Wolsey, and Fisher 1978). However, we show a negative result about the submodularity of $v^*(S_R)$ and $\hat{v}(S_R)$, which precludes the use of such an approach. Recall that a set function $f : 2^\Omega \to \mathbb{R}_+$ on groundset Ω is submodular if $f(U \cup \{i\}) - f(U) \ge f(W \cup \{i\}) - f(W)$ for all $U \subseteq W \subseteq \Omega$ and $i \in \Omega \setminus W$.

Proposition 2. *The objective functions $v^*(S_R)$ and $\hat{v}(S_R)$ are not submodular functions.*

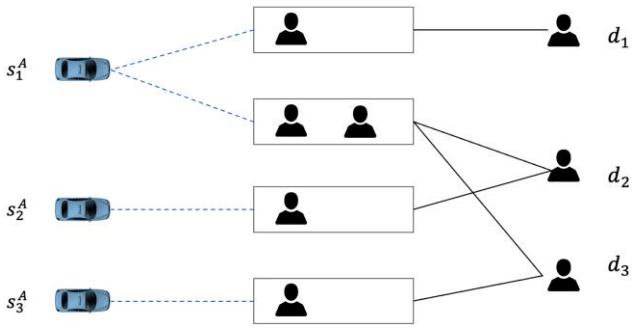
Proof. Recall that the ground set for both functions v^* and $\hat{v}$ is $\Omega := S_A$ the set of augmented vehicles. We provide an SRAMF instance with $n = 1$ scenario where these functions are not submodular. Consider a shareability graph with $|S_A| = 3$, $S_B = \emptyset$ and three ride requests $\{d_1, d_2, d_3\}$. Let $p = 3$, that is, each vehicle can carry at most two requests. The set of hyperedges is

$$\{(s_1^A, d_1), (s_1^A, d_2, d_3), (s_2^A, d_2), (s_3^A, d_3)\}.$$

See also Figure 4. The value of each hyperedge reduces to the number of ride requests it covers.

Let subsets $U = \{s_1^A\}$ and $W = \{s_1^A, s_2^A\}$. Also, let $i = s_3^A$. Clearly, $v^*(U) = 2$ (serving d_2, d_3), $v^*(W) = 2$ (serving d_1, d_2 or d_2, d_3), $v^*(U \cup \{i\}) = 2$ (serving d_1, d_3 or d_2, d_3),

Figure 4. (Color online) Nonsubmodularity of Function $v^*(S_R)$



and $v^*(W \cup \{i\}) = 3$. Therefore, we have

$$v^*(W \cup \{i\}) - v^*(W) = 1 > 0 = v^*(U \cup \{i\}) - v^*(U),$$

which implies the set function v^* is not submodular. It is easy to check that the LP value function $\hat{v} = v^*$ for this instance, so function $\hat{v}$ is also not submodular. □

4. Approximation Algorithms for SRAMF

This section provides two different approximation algorithms for SRAMF. Both the algorithms focus on solving the fractional assignment problem (7) and achieve approximation ratios $1/p$ and $\approx (e-1)/(2e \cdot \ln p)$, respectively. Combined with Theorem 1, these imply approximation algorithms for SRAMF with an additional factor of $1/p$.

4.1. Local Search Algorithm for Mid-Capacity SRAMF

The mid-capacity SRAMF models the current ride-hailing market, in which each vehicle can deliver two to four ride requests simultaneously. In this section, we propose an LSLPR algorithm that obtains $1/p$-approximation for the fractional assignment problem (7).

4.1.1. Overview of the LSLPR Algorithm. Let $\epsilon > 0$ be an arbitrarily small parameter to serve as the algorithm's stopping criterion. The outline of the LSLPR algorithm is as follows:

1. Start from any solution $S_R \subseteq S_A$ with $|S_R| = K$.
2. Consider all alternative solutions $S_{R'} = S_R - \{i\} + \{i'\}$ where $i \in S_R$ and $i' \notin S_R$ after swapping one vehicle and evaluate the corresponding LP value $\hat{v}(S_{R'})$.
3. Change the current solution S_R to $S_{R'}$ if the objective value improves significantly, i.e., $\hat{v}(S_{R'}) > (1+\epsilon) \cdot \hat{v}(S_R)$.
4. Stop if such a significant local swap does not exist.

Formally, let k index the iterations. Let S_R^k denote the current solution in iteration k. The following subroutine implements a single iteration.

Algorithm 1 (Local Swap Subroutine)

for $i \in S_R^k$ *and* $i' \in S_A \backslash S_R^k$ **do**
  obtain $\hat{v}(S_R^k - i + i')$ by solving the fractional assignment problem;
end
let (c, c') be the pair that maximizes $\hat{v}(S_R^k - i + i')$ over $i \in S_R^k$ and $i' \in S_A \backslash S_R^k$;
if $\hat{v}^*(S_R^k - c + c') > (1+\epsilon) \cdot \hat{v}(S_R^k)$ **then**
  set $S_R^{k+1} \leftarrow S_R^k - c + c'$ and continue with $k \leftarrow k+1$;
else
  halt local search and output S_R^k;
end

In a broad sense, the local swap subroutine does not necessarily enumerate all pairs (i, i') to search for the optimal (c, c'). A more efficient alternative is terminating each iteration at the first pair of $i \in S_R$ and $i' \in S_A \backslash S_R$ that increases the objective by more than $\epsilon \cdot \hat{v}(S_R^k)$.

The complete LSLPR algorithm is as follows.

Algorithm 2 (LSLPR Algorithm for Midcapacity SRAMF)

Data: Augmented supply S_A, basis supply S_B, scenarios $\{\xi_\ell\}_{\ell=1}^N$ and $\epsilon > 0$.
Result: Near-optimal $S_R \subset S_A$ and the corresponding trip assignment.
Initialization: Set $k = 1$ and randomly select K vehicles from S_A as S_R^1;
while $k \le k_{\max}$ **do**
  Run the local swap subroutine in Algorithm 1;
  Obtain the final trip assignment with $S_R = S_R^{k_{\max}}$ using the greedy algorithm (Theorem 1).
end

Algorithm 2 obtains the final selection of vehicles $S_R^{k_{\max}}$, where the maximal number of iterations k_{max} will be derived later. In the final step, the algorithm obtains an *integral* assignment for each scenario instead of the fractional assignments. To this end, we can use the greedy algorithm (see Theorem 1) to select the assignment for each scenario, which is guaranteed to have an objective value at least $1/p$ times the fractional assignment. In Section 4.1.2, we first analyze the approximation ratio and then the computational complexity of LSLPR.

4.1.2. Analysis of the LSLPR Algorithm. Recall that S_R is the solution obtained by our algorithm and $|S_R| = K$. Let S_O denote the optimal solution: We assume (without loss of generality) $|S_O| = K$. Note that S_O is a fixed subset only used in the analysis. Also, let $\boldsymbol{x} = \langle \boldsymbol{x}^\xi \rangle$ and $\boldsymbol{z} = \langle \boldsymbol{z}^\xi \rangle$ denote the optimal LP solutions to $\hat{v}(S_R)$ and $\hat{v}(S_O)$, respectively.

It will be convenient to consider the overall hypergraph on vertices $S_A \cup S_B \cup (\cup_\xi D(\xi))$ and hyperedges $\cup_\xi E(\xi)$. As the objective $\hat{v}(\cdot)$ is additive over the scenarios ξ, we may assume, by duplicating demands and hyperedges (if necessary), that demands $D(\xi)$ and hyperedges $E(\xi)$ are disjoint across scenarios ξ. Recall that $\boldsymbol{x}^\xi$ (and $\boldsymbol{z}^\xi$) has a decision variable corresponding to each hyperedge in $E(\xi)$. For each demand $d \in \cup_\xi D(\xi)$, let H_d denote the hyperedges incident to it. For each vehicle $i \in S_A \cup S_B$ and scenario ξ, let $E_{i,\xi}$ denote the hyperedges in $E(\xi)$ containing i. Therefore, $F_i := \cup_\xi E_{i,\xi}$ is the set of hyperedges incident to vehicle i.

For any demand d, the following lemma sets up a mapping between the hyperedges (incident to d) used in the solutions $\boldsymbol{x}$ and $\boldsymbol{z}$. For the analysis, we add a dummy hyperedge $\perp$ incident to d so that the assignment constraints in the LP solutions $\boldsymbol{x}$ and $\boldsymbol{z}$ are binding at d. So, $\sum_{e \in H_d} x_e + x_\perp = 1$ and $\sum_{f \in H_d} z_f + z_\perp = 1$. Let $H'_d := H_d \cup \{\perp\}$ denote the hyperedges incident to d.

Lemma 1. *For any demand d, there exists a decomposition mapping $\Delta_d : H'_d \times H'_d \to \mathbb{R}$ satisfying the following conditions:*

1. *Mapping $\Delta_d(e,f) \ge 0$ for all $e, f \in H'_d$;*
2. *For all $f \in H'_d$ $\sum_{e \in H'_d} \Delta_d(e,f) = z_f$;*
3. *For all $e \in H'_d$ $\sum_{f \in H'_d} \Delta_d(e,f) = x_e$.*

Figure 5. Mapping $\Delta_d(e,f)$ with $E(\xi) = \{e_1, e_2, e_3\}$

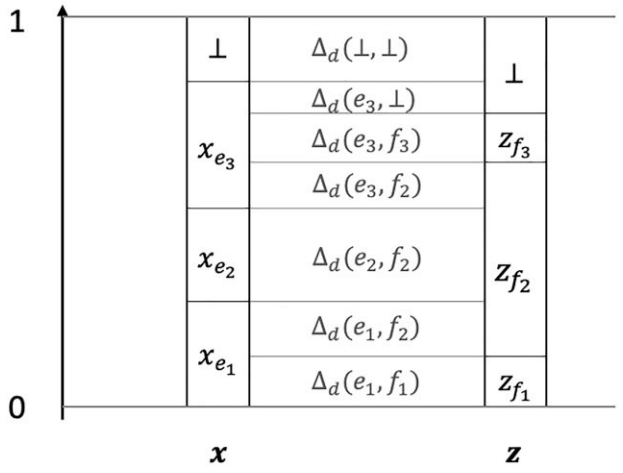


Figure 5 illustrates this mapping. Appendix C includes the definition of $\Delta_d(e,f)$ and the proof of Lemma 1. Note that $\sum_{e\in H'_d}\sum_{f\in H'_d}\Delta_d(e,f) = 1$ for any demand d. For any subset $F \in H'_d$, we use the shorthand $\Delta_d(e,F) := \sum_{f\in F}\Delta_d(e,f)$ and $\Delta_d(F,e) := \sum_{f\in F}\Delta_d(f,e)$.

Here is an outline of the remaining analysis. Let $\mathcal{L}$ denote a bijection between S_R (LSLPR algorithm's solution) and S_O (optimal solution), consisting of pairs (i_1, i_2) where $i_1 \in S_R$ and $i_2 \in S_O$. We also ensure that $\mathcal{L}$ contains the pairs (i, i) for all vehicles $i \in S_R \cap S_O$. There is such a bijection because $|S_R| = K = |S_O|$. We first consider a swap $S_R - \{i_1\} + \{i_2\}$ where $(i_1, i_2) \in \mathcal{L}$, and lower bound the objective increase. The approximate local optimality of S_R implies that the objective increase is at most $\epsilon \cdot \hat{v}(S_R)$. Then, we add the inequalities corresponding to the objective increase for the swaps in $\mathcal{L}$ and obtain the approximation ratio.

4.1.2.1. Analysis of a Single Swap (i_1, i_2).

Consider any $i_1 \in S_R$ and $i_2 \in S_O$. We now lower bound $\hat{v}(S_R - \{i_1\} + \{i_2\}) - \hat{v}(S_R)$. Recall that for any subset S, $\hat{v}(S) = \frac{1}{N}\sum_\xi \hat{v}(S,\xi)$ where $\hat{v}(S,\xi)$ is the LP value for scenario ξ. Therefore, we have

$$\hat{v}(S_R - \{i_1\} + \{i_2\}) - \hat{v}(S_R) = \frac{1}{N}\sum_\xi (\hat{v}(S_R - \{i_1\} + \{i_2\}, \xi) - \hat{v}(S_R, \xi)).$$

We now focus on a single scenario ξ and lower bound $\hat{v}(S_R - \{i_1\} + \{i_2\}, \xi) - \hat{v}(S_R, \xi)$. $\bar{\boldsymbol{x}}^\xi$ represents a feasible solution for fractional assignment $\hat{v}(S_R - \{i_1\} + \{i_2\}, \xi)$. Recall that $\boldsymbol{x}^\xi$ denotes the optimal solution for LP $\hat{v}(S_R, \xi)$. Therefore, we can then bound:

$$\hat{v}(S_R - \{i_1\} + \{i_2\}, \xi) - \hat{v}(S_R, \xi) \geq \boldsymbol{v}^\top \bar{\boldsymbol{x}}^\xi - \boldsymbol{v}^\top \boldsymbol{x}^\xi, \tag{8}$$

where $\boldsymbol{v}$ is the vector of hyperedge values for $E(\xi)$. As we focus on a single scenario ξ, we drop ξ from the notation whenever it is clear.

We are now ready to construct the new fractional assignment $\bar{\boldsymbol{x}}$. Define the following:

1. Let $\bar{x}_e = 0$ for all $e \in F_{i_1}$. This corresponds to dropping vehicle i_1 from S_R.
2. Let $\bar{x}_e = z_e$ for all $e \in F_{i_2}$. This corresponds to adding vehicle i_2 to S_R.
3. Let $\bar{x}_e = x_e - \max_{d\in e}\Delta_d(e, F_{i_2} \cap H_d)$ for all $e \in E(\xi) \setminus F_{i_1} \setminus F_{i_2}$.

If $i_1 = i_2$, then we drop case 1. The third case is needed to make space for the hyperedges incident to the new vehicle i_2 (which is increased in case 2). The following two lemmas prove the feasibility of this solution $\bar{\boldsymbol{x}}$ and bound its objective value. Later, we assume that $i_1 \neq i_2$ (the proof for $i_1 = i_2$ is nearly the same, in fact even simpler). Therefore, $i_1 \in S_R \setminus S_O$ and $i_2 \in S_O \setminus S_R$.

Lemma 2. *The fractional assignment $\bar{\boldsymbol{x}}$ is a feasible solution for $\hat{v}(S_R - \{i_1\} + \{i_2\})$.*

Proof for Lemma 2. We show the feasibility by checking all constraints in (4). Note that $\bar{x}_e = 0$ for all hyperedges e incident to a vehicle in $S_A \setminus (S_R - \{i_1\} + \{i_2\})$.

Constraint $\bar{\boldsymbol{x}} \geq 0$. It suffices to check this for hyperedges $e \in E \setminus F_{i_1} \setminus F_{i_2}$. Note that

$$\bar{x}_e = x_e - \max_{d\in e}\Delta_d(e, F_{i_2} \cap H_d) = \min_{d\in e}(x_e - \Delta_d(e, F_{i_2} \cap H_d)) \geq 0,$$

where the inequality uses Lemma 1 (condition 3), that is, $x_e = \Delta_d(e, H'_d) \geq \Delta_d(e, F_{i_2} \cap H_d)$.

Constraint (4a): By definition of $\bar{\boldsymbol{x}}$, for any demand d, we have

$$\begin{aligned}
\sum_{e\in H_d}\bar{x}_e &\leq \sum_{e\in H_d\cap F_{i_2}} z_e + \sum_{e\in H_d\setminus F_{i_1}\setminus F_{i_2}} [x_e - \Delta_d(e, F_{i_2}\cap H_d)] \\
&\leq \sum_{e\in H_d\cap F_{i_2}} z_e + \sum_{e\in H'_d}[x_e - \Delta_d(e, F_{i_2}\cap H_d)] \qquad (9) \\
&= \sum_{e\in H_d\cap F_{i_2}} z_e + \sum_{e\in H'_d} x_e - \sum_{e\in H'_d}\Delta_d(e, F_{i_2}\cap H_d) \\
&= \sum_{e\in H_d\cap F_{i_2}} z_e + \sum_{e\in H'_d} x_e - \sum_{f\in F_{i_2}\cap H_d}\Delta_d(H'_d, f) \\
&= \sum_{e\in H'_d} x_e = 1. \qquad (10)
\end{aligned}$$

(9) uses $x_e \geq \Delta_d(e, F_{i_2} \cap H_d)$ by Lemma 1, and the first equality in (10) uses $z_f = \Delta_d(H'_d, f)$ by Lemma 1 (condition 2).

Constraint (4b): The augmented vehicle set can be divided into three groups.

1. Vehicle i_1: $\sum_{e\in F_{i_1}}\bar{x}_e = 0$.
2. Vehicle i_2: $\sum_{e\in F_{i_2}}\bar{x}_e = \sum_{e\in F_{i_2}} z_e \leq 1$ by definition.
3. Vehicles $j \neq i_1, i_2$: $\sum_{e\in F_j}\bar{x}_e \leq \sum_{e\in F_j} x_e \leq 1$. Here, we used the definition of $\bar{x}_e$ and $\Delta_d(\cdot,\cdot) \geq 0$ by Lemma 1 (condition 1).

Therefore, $\bar{\boldsymbol{x}}$ is a feasible fractional assignment solution. □

Lemma 3. *The increase in the objective is*

$$\sum_{e\in E(\xi)} v_e(\overline{x}_e^{\xi} - x_e^{\xi}) \geq \sum_{e\in F_{i_2}\cap E(\xi)} v_e z_e^{\xi} - \sum_{f\in F_{i_1}\cap E(\xi)} v_f x_f^{\xi} - \sum_{e\in E(\xi)} v_e \sum_{d\in e} \Delta_d(e, F_{i_2}\cap H_d).$$

Proof for Lemma 3. By definition of $\bar{x}$,

$$\overline{x}_e - x_e = \begin{cases} z_e & \text{if } e\in F_{i_2} \\ -x_e & \text{if } e\in F_{i_1} \\ -\max\limits_{d\in e} \Delta_d(e, F_{i_2}\cap H_d) & \text{otherwise.} \end{cases}$$

We used that $x_e = 0$ for all $e \in F_{i_2}$ as $i_2 \in S_A\backslash S_R$. Therefore, we have

$$\begin{aligned}\sum_{e\in E(\xi)} v_e(\overline{x}_e - x_e) &\geq \sum_{e\in F_{i_2}\cap E(\xi)} v_e z_e - \sum_{f\in F_{i_1}\cap E(\xi)} v_f x_f - \sum_{e\in E(\xi)} v_e \max_{d\in e} \Delta_d(e, F_{i_2}\cap H_d) \\ &\geq \sum_{e\in F_{i_2}\cap E(\xi)} v_e z_e - \sum_{f\in F_{i_1}\cap E(\xi)} v_f x_f - \sum_{e\in E(\xi)} v_e \sum_{d\in e} \Delta_d(e, F_{i_2}\cap H_d). \quad \square\end{aligned}$$

Combining Lemmas 2 and 3, and adding over scenarios ξ, we obtain the following.

Lemma 4. *For any pair $(i_1, i_2) \in \mathcal{L}$, we have*

$$\hat{v}(S_R - \{i_1\} + \{i_2\}) - \hat{v}(S_R) \geq \sum_{e\in F_{i_2}} v_e z_e - \sum_{f\in F_{i_1}} v_f x_f - \sum_{e\in E} v_e \sum_{d\in e} \Delta_d(e, F_{i_2}\cap H_d).$$

4.1.2.2. Combining All the Swaps.

Recall that $\mathcal{L}$ is a bijection between S_R and S_O, so $|\mathcal{L}| = K$. Moreover, using the local-search termination condition, there is no swap that improves the objective of the final solution S_R by more than $\epsilon\cdot\hat{v}(S_R)$. Hence,

$$\begin{aligned} K\epsilon\cdot\hat{v}(S_R) &\geq \sum_{(i_1,i_2)\in\mathcal{L}} [\hat{v}(S_R - \{i_1\} + \{i_2\}) - \hat{v}(S_R)] \\ &\geq \sum_{(i_1,i_2)\in\mathcal{L}} \left[\sum_{e\in F_{i_2}} v_e z_e - \sum_{f\in F_{i_1}} v_f x_f - \sum_{e\in E} v_e \sum_{d\in e} \Delta_d(e, F_{i_2}\cap H_d)\right] \qquad (11) \\ &= \sum_{i_2\in S_O}\sum_{e\in F_{i_2}} v_e z_e - \sum_{i_1\in S_R}\sum_{f\in F_{i_1}} v_f x_f - \sum_{i_2\in S_O}\sum_{e\in E} v_e \sum_{d\in e} \Delta_d(e, F_{i_2}\cap H_d) \\ &\geq \sum_{i_2\in S_O}\sum_{e\in F_{i_2}} v_e z_e - \sum_{i_1\in S_R}\sum_{f\in F_{i_1}} v_f x_f - \sum_{e\in E} v_e \sum_{d\in e} \Delta_d(e, H_d) \qquad (12) \\ &\geq \sum_{i_2\in S_O}\sum_{e\in F_{i_2}} v_e z_e - \sum_{i_1\in S_R}\sum_{f\in F_{i_1}} v_f x_f - \sum_{e\in E} v_e \sum_{d\in e} x_e \qquad (13) \\ &= v^T z - v^T x - \sum_{e\in E} |\{d\in e\}| v_e x_e \\ &\geq v^T z - v^T x - (p-1) v^T x = v^T z - p\cdot v^T x = \hat{v}(S_O) - p\cdot\hat{v}(S_R). \qquad (14)\end{aligned}$$

(11) is by Lemma 4, (12) uses that $\{F_{i_2}\}_{i_2\in S_O}$ are disjoint, (13) uses Lemma 1, and the inequality in (14) uses that each hyperedge has at most $p-1$ demands.

Setting $\epsilon = 1/pK^2$, it follows that $\hat{v}(S_R) \geq 1/(p + o(1))\cdot\hat{v}(S_O)$. Combined with Theorem 1, we obtain $v^*(S_R) \geq 1/p\cdot\hat{v}(S_R) \geq (1/(p^2 + o(p)))\cdot\hat{v}(S_O)$.

Theorem 2. *The LSLPR algorithm for SRAMF is a $1/p^2$-approximation algorithm.*

4.1.2.3. Time Complexity of the LSLPR Algorithm.

Each iteration of Algorithm 1 involves considering $K(n_A - K)$ potential swaps and recall that $n_A = |S_A|$. For each swap, we need to evaluate $\hat{v}$, which can be done using any polynomial time LP algorithm such as the ellipsoid method (Bertsimas and Tsitsiklis 1997). Therefore, the time taken in each iteration is polynomial.

We now bound the number of local search iterations. In each iteration, the objective value increases by a factor of at least $1+\epsilon$. Therefore, after k iterations,

$$\hat{v}(S_R^{k+1}) \geq (1+\epsilon)^k \hat{v}(S_R^1).$$

Clearly, the assignment associated with the initial solution S_R^0 has a lower bound $\hat{v}(S_R^0) \geq (1/N)\cdot v_{\min}$, where $v_{\min} = \min_{e:v_e>0} v_e$ is the minimum value over all hyperedges. Recall that hyperedges with nonpositive values are not considered in any assignment. The maximum objective of any solution is at most $(n_A + n_B)\cdot v_{\max}$, where $|S_A| = n_A$, $|S_B| = n_B$ and $v_{\max} = \max_e v_e$ is the maximum value over all hyperedges. Hence,

$$(n_A + n_B)\cdot v_{\max} \geq \hat{v}(S_R^{k+1}) \geq (1+\epsilon)^k \cdot \frac{1}{N} v_{\min},$$

which implies that the maximum number of iterations

$$\begin{aligned} k_{\max} &\leq \log_{1+\epsilon}\left(\frac{N(n_A+n_B)v_{\max}}{v_{\min}}\right) \\ &= O\left(\frac{1}{\epsilon}\log\frac{N(n_A+n_B)v_{\max}}{v_{\min}}\right).\end{aligned}$$

Using $\epsilon = 1/pK^2$, it follows that the number of iterations is polynomial.

The last step of Algorithm 2 implements the greedy p-set packing algorithm for each scenario, which also takes polynomial time. It follows that LSLPR solves the SRAMF problem in polynomial time regarding parameters $p, K, N, |E|, n_A$, and n_B.

4.2. Max-Min Online Algorithm for High-Capacity SRAMF

The LSLPR algorithm is capable of assigning rides in shared mobility applications using midcapacity vehicles. When the maximal capacity of vehicles in MoD is large (e.g., the maximum capacity of MoD transit service is 10 in Alonso-Mora et al. (2017)), the $1/p^2$-approximation ratio is disadvantageous. We propose an alternative method for high-capacity SRAMF. The main idea of the max-min online (MMO) algorithm is to use LP-duality to reformulate $\hat{v}$ as a *covering* linear program. Then, the max-min optimization in Feige et al. (2007) can further improve the approximation ratio. This framework requires two technical properties (monotonicity and online competitiveness), which are satisfied in the SRAMF problem. We will prove that the MMO algorithm obtains an approximation ratio of $(1-(1/e))(1/2p\ln p)$.

Using LP duality and the definition of $\hat{v}(S_R)$ (see the derivation in Appendix C, Section C.3), we can reformulate

$$\begin{aligned}\hat{v}(S_R) = \underset{u}{\text{minimize}} \quad & \sum_{\xi}\sum_{g\in G} u_{g,\xi} \qquad (15)\\ \text{s.t.} \quad & \sum_{g\in e} u_{g,\xi} \ge \frac{v_e}{N},\\ & \forall e \in F_{i,\xi}, \quad \forall \xi, \quad \forall i \in S_R \cup S_B,\\ & \boldsymbol{u} \ge 0.\end{aligned}$$

Here, $G = S_A \cup S_B \cup (\cup_\xi D(\xi))$ is a combined groundset consisting of all vehicles and demands from all scenarios. For any vehicle i and scenario ξ, set $F_{i,\xi} \subseteq E(\xi)$ denotes all the hyperedges incident to i in scenario ξ.

We can scale the covering constraints to normalize the right-hand side to one and rewrite the constraints as $\sum_{g\in e}\frac{N}{v_e}u_{g,\xi} \ge 1$. The *row sparsity* of this constraint matrix (i.e., the maximum number of nonzero entries in any constraint) is $\max_{e\in E}|e| = p$ and $v_e > 0$ for all hyperedges. Let $\boldsymbol{c}_e$ be the row of constraint coefficients for any hyperedge $e \in E = \cup_\xi E(\xi)$, that is,

$$c_e(g,\xi) = \begin{cases} \dfrac{N}{v_e} & \text{if } g \in e \text{ and } e \in E(\xi)\\ 0 & \text{otherwise}\end{cases}.$$

Then, the SRAMF problem with fractional assignments $\max_{S_R\subseteq S_A:|S_R|\le K}\hat{v}(S_R)$ can be treated as the following max-min problem:

$$\max_{S_R\subseteq S_A:|S_R|\le K} \min_{u}\{\mathbf{1}^\top \boldsymbol{u} \,|\, \boldsymbol{c}_e^\top \boldsymbol{u} \ge 1,\ \forall e\in F_i,\ \forall i \in S_R\cup S_B;\ \boldsymbol{u}\ge 0\}, \qquad (16)$$

where $F_i = \cup_\xi F_{i,\xi}$ for each vehicle i. For the remainder, $t = 1, 2, \ldots$ indexes steps of the online algorithm.

The main result is as follows.

Theorem 3. *There is a* $(e-1)/(2e+o(1))\ln p$*-approximation algorithm for* (16).

Before proving this result, we introduce two important properties.

Definition 1 (Competitive Online Property). An α-competitive online algorithm for the covering problem (15) takes as input any sequence $(i_1, i_2, \ldots, i_t, \ldots)$ of vehicles from S_A and maintains a nondecreasing solution $\boldsymbol{u}$ such that the following hold for all steps t.

- Solution $\boldsymbol{u}$ satisfies constraints $\boldsymbol{c}_e^\top \boldsymbol{u} \ge 1$ for $e \in F_i$, for all vehicles $i \in \{i_1, i_2, \ldots, i_t\}$, and
- Solution $\boldsymbol{u}$ is an α-approximate solution, i.e., the objective $\mathbf{1}^\top \boldsymbol{u} \le \alpha \cdot \hat{v}(\{i_1, i_2, \ldots, i_t\})$.

The online algorithm may only increase variables $\boldsymbol{u}$ in each step t.

Definition 2 (Monotone Property). For any $\boldsymbol{u} \ge 0$ and $S \subseteq S_A$, let

$$Aug^*(S|\boldsymbol{u}) := \{\min_{w\ge 0} \mathbf{1}^\top \boldsymbol{w} : \boldsymbol{c}_e^\top(\boldsymbol{u}+\boldsymbol{w}) \ge 1, \quad \forall e \in F_i,\ \forall i \in S \cup S_B\}.$$

The covering problem (15) is said to be monotone if for any $\boldsymbol{u} \ge \boldsymbol{u}' \ge 0$ (coordinate wise) and any $S \subseteq S_A$, $Aug^*(S|\boldsymbol{u}) \le Aug^*(S|\boldsymbol{u}')$.

These properties were used by Feige et al. (2007) to show the following result.

Theorem 4 (Feige et al. 2007). *If the covering problem* (15) *satisfies the monotone and* α*-competitive online properties, there is a* $(e-1)/(e\cdot\alpha)$*-approximation for the max-min problem in* (16).

Our max-min problem indeed satisfies both these properties.

Lemma 5. *The covering problem* (15) *has an* $\alpha = O(\ln p)$ *competitive online algorithm. Moreover, when* p *is large, the factor* $\alpha = (2+o(1))\ln p$.

Proof. Recall that (15) is a covering LP with row-sparsity p. Moreover, in the online setting, constraints to (15) arrive over time. Therefore, this is an instance of online covering LPs, for which an $O(\ln p)$-competitive algorithm is known (Gupta and Nagarajan 2014). See also Buchbinder et al. (2014) for simpler proof. Moreover, one can optimize the constant factor in Buchbinder et al. (2014) to get $\alpha = (2+o(1))\ln p$.

These previous papers work with the online model when only one covering constraint arrives in each step. Although Lemma 5 involves multiple covering constraints F_i arriving in each step, this complexity can easily be reduced to the prior setting as follows. We introduce the constraints in F_i one by one in any order. The algorithms in Gupta and Nagarajan (2014) and Buchbinder et al. (2014) can therefore be used directly. □

Lemma 6. *The covering problem* (15) *is monotone.*

Proof. Consider any $\boldsymbol{u} \geq \boldsymbol{u}' \geq 0$ and any $S \subset S_A$. Let $\boldsymbol{w}' \geq 0$ denote an optimal solution to $Aug^*(S_R|\boldsymbol{u}')$. As all constraint coefficients $\boldsymbol{c}_e \geq 0$, it follows that $\boldsymbol{c}_e^\top(\boldsymbol{u}+\boldsymbol{w}') \geq \boldsymbol{c}_e^\top(\boldsymbol{u}'+\boldsymbol{w}') \geq 1$ for all $e \in F_i$ and $i \in S \cup S_B$. Hence, $\boldsymbol{w}'$ is also a feasible for the constraints in $Aug^*(S|\boldsymbol{u})$. Therefore, $Aug^*(S|\boldsymbol{u}) \leq \mathbf{1}^\top \boldsymbol{w}' = Aug^*(S|\boldsymbol{u}')$, which proves the monotonicity. □

Combining Lemmas 5 and 6 with Theorem 4, we obtain Theorem 3. Our $\Omega(1/\ln p)$ approximation ratio is nearly the best possible for the max-min problem (16), as the problem is hard to approximate to a factor better than $O(\ln\ln p/\ln p)$ (Feige et al. 2007).

We now describe the complete algorithm for SRAMF. This is a combination of the online LP algorithm from Buchbinder et al. (2014) and the max-min algorithm from Feige et al. (2007). For any ordered subset S of vehicles, let $\hat{v}_{ON}(S)$ denote the objective value of the online algorithm for (15) after adding constraints corresponding to the vehicles in S (in that order). Algorithm 3 describes the updates performed by the online algorithm when a vehicle i is added.

Algorithm 3 (Updating Subroutine in the MMO Algorithm)
For a given $i \in S_A \cup S_B$, perform the following updates;
for $e \in F_i = \cup_\xi F_{i,\xi}$ **do**
  let $\{u^-_{g,\xi}\}_{g\in e}$ be the values of variables in hyperedge e and $\Gamma^-_e = \sum_{g\in e} u^-_{g,\xi}$;
  if $\Gamma^-_e < \frac{v_e}{N}$ **then**
    update $u_{g,\xi} \leftarrow \left(u^-_{g,\xi} + \frac{v_e}{N}\delta\right) \cdot \frac{1+|e|\cdot\delta}{\frac{N}{v_e}\Gamma^-_e + |e|\cdot\delta} - \frac{v_e}{N}\delta$, for all $g \in e$.
  end
end

Proof for the Updating Subroutine in the MMO Algorithm. Consider the updates when vehicle i is added. Consider any scenario ξ and hyperedge $e \in F_{i,\xi}$: the corresponding covering constraint is $c_e^T u = (N/v_e)\sum_{g\in e} u_{g,\xi} \geq 1$. Let τ be a continuous variable denoting time and $\delta > 0$ be a constant. The online LP algorithm in (Buchbinder et al. 2014) raises variables $u_{g,\xi}$ in a continuous manner as follows:

$$\frac{\partial u_{g,\xi}}{\partial \tau} = \frac{N}{v_e} u_{g,\xi} + \delta, \qquad \forall g \in e, \tag{17}$$

until the constraint is satisfied. Letting $\Gamma_e = \sum_{g\in e} u_{g,\xi}$, we have

$$\frac{\partial \Gamma_e}{\partial \tau} = \frac{N}{v_e}\sum_{g\in e} u_{g,\xi} + |e|\cdot\delta = \frac{N}{v_e}\Gamma_e + |e|\cdot\delta.$$

By integrating, it follows that the duration of this update is

$$T = \int_{\Gamma=\Gamma^-_e}^{\Gamma^+_e} \frac{\partial \Gamma_e}{\frac{N}{v_e}\Gamma_e + |e|\cdot\delta} = \frac{v_e}{N}\cdot\ln\left(\frac{\frac{N}{v_e}\Gamma^+_e + |e|\cdot\delta}{\frac{N}{v_e}\Gamma^-_e + |e|\cdot\delta}\right)$$
$$= \frac{v_e}{N}\cdot\ln\left(\frac{1+|e|\cdot\delta}{\frac{N}{v_e}\Gamma^-_e + |e|\cdot\delta}\right).$$

Earlier, Γ^-_e and Γ^+_e denote the values of Γ_e at the start and end of this update step; $\Gamma^+_e = v_e/N$ as the updates stop as soon as the constraint is satisfied. For each $g \in e$, using (17),

$$T = \int_{\tau=0}^{T} \frac{\partial u_{g,\xi}}{\frac{N}{v_e}u_{g,\xi} + \delta} = \frac{v_e}{N}\cdot\ln\left(\frac{\frac{N}{v_e}u^+_{g,\xi} + \delta}{\frac{N}{v_e}u^-_{g,\xi} + \delta}\right).$$

Again, $u^-_{g,\xi}$ and $u^+_{g,\xi}$ denote the values of $u_{g,\xi}$ at the start and end of this update step. Combined with the previous value for T, we get a closed-form expression for the new variable values:

$$\frac{N}{v_e}u^+_{g,\xi} + \delta = \left(\frac{N}{v_e}u^-_{g,\xi} + \delta\right)\cdot\frac{1+|e|\cdot\delta}{\frac{N}{v_e}\Gamma^-_e + |e|\cdot\delta}, \qquad \forall g \in e. \quad \square$$

The complete MMO algorithm is described in Algorithm 4.

Algorithm 4 (MMO Algorithm for SRAMF)
Data: Augmented supply S_A, basis supply S_B, hypergraph G with $E(\xi)$, and $\epsilon > 0$.
Result: Near-optimal $S_R \subset S_A$ and the corresponding trip assignment.
Initialization: $S_R \leftarrow \emptyset$ and dual variables $\mathbf{u} \leftarrow 0$;
For each vehicle in S_B (in any order), run Algorithm 3 to obtain $\hat{v}_{ON}(S_B)$
for $k = 1, \ldots, K$ **do**
  for $i \in S_A \backslash S_R$ **do**
    Run the updating subroutine in Algorithm 3 and obtain $\hat{v}_{ON}(S_B + S_R + \{i\})$.
  end
  $i^* = \arg\max_{i\in S_A\backslash S_R} \hat{v}_{ON}(S_B + S_R + \{i\})$;
  $S_R \leftarrow S_R + \{i^*\}$;
end

4.3. Extensions to SRAMF Under Partition Constraints

We now consider a more general setting where the augmented set S_A is partitioned into M subsets $S_A(1)$, $\cdots$ $S_A(m)$, $\cdots$, $S_A(M)$, and the platform requires K_m vehicles from each subset.

Example 3. In the market segmentation of Example 1 described in Section 1, there are M types of vehicles, so the cardinality constraint is further specified for each vehicle type as partition constraints $\sum_{i\in S_A(m)} y_i \leq K_m$ for all $m \in [M]$.

Example 4. In the mixed autonomy application of Example 2, there are M separate AV zones, and the repositioning capacity requirement is proportional to the demand density in each zone; $\sum_{i\in S_A(m)} y_i \le K_m$ is now a constraint for each AV zone $m \in [M]$.

The original SRAMF problem (2) is now expanded to solve

$$\underset{y}{\text{maximize}} \quad \mathbf{E}[Q(y,\xi)] \tag{18}$$

$$\text{s.t.} \sum_{i\in S_A(m)} y_i \le K_m \qquad \forall m \in [M], \tag{18a}$$

$$y_i \in \{0,1\} \quad \forall i \in S_A. \tag{18b}$$

We can extend our result to obtain the following.

Theorem 5. *The MMO algorithm is a* $(1/(4+o(1))p\log p)$*-approximation algorithm for SRAMF with partition constraints.*

The proof is identical to that of Theorem 3. The only difference is the use of the following result for max-min covering under a partition constraint (instead of Theorem 4, which only holds for a cardinality constraint).

Theorem 6 (Gupta, Nagarajan, and Ravi 2015). *If the covering problem* (15) *satisfies the monotone and* α*-competitive online properties, there is a* $1/2\alpha$*-approximation for the max-min problem with a partition (or matroid) constraint.*

5. Numerical Experiments

5.1. Data Description and Experiment Setup

We evaluate the effectiveness of the proposed approximation algorithms in two hypothetical mixed-fleet scenarios:

1. **Setting 1** simulates mixed fleets of standard and premium vehicles in Example 1 and represents the midcapacity SRAMF scenario. The MoD platform periodically repositions premium vehicles of S_A to serve ride requests when the future demand exceeds the capacity of standard vehicles in S_B. We consider the stochastic nature of system dynamics as the probability of demand surges in some zones across the city. The value of hyperedges in these zones increases when demand surges, rewarding algorithms that successfully reposition in locations with high surge probability. The main task is to reposition premium vehicles to accommodate predicted surge demand.

2. **Setting 2** simulates the early deployment of AVs in Example 2 and represents the high-capacity SRAMF scenario. Because of regulatory or technological restrictions, we assume that automated MoD buses operate only within certain AV zones (Chen et al. 2017b) and deliver up to 10 passengers per trip (Alonso-Mora et al. 2017). The main task is to periodically reposition K automated MoD buses in these AV zones to accommodate future demand.

To demonstrate the value of using a stochastic assignment framework, we consider two benchmark models:

1. **Benchmark 1:** *Stochastic assignment using IP solver* solves SRAMF exactly using the SAA approach in Section 3.2. This benchmark method and approximation algorithms use the same set of samples to assess on a fair basis. The SAA approach is implemented in a state-of-the-art IP solver (Gurobi 9.1).

2. **Benchmark 2:** *Assignment with mean demand forecasts* solves a deterministic ride-pooling assignment problem based on the mean demand forecasts. This method solves the joint vehicle repositioning and trip assignment problem using a one-shot approach based on the mean hyperedge value and demand distribution $F(\xi)$. The goal is to address the significance of considering demand and supply uncertainties in SRAMF, albeit at the expense of increased computing complexity.

The objective values of Benchmark 2 and SRAMF are not comparable. The following reconstruction procedure is therefore used in evaluations. (1) Select a set of augmented supply S_R based on the average scenario. (2) Generate a new set of test samples as outlined in the SAA method. (3) Recompute the objective values for all algorithms using identical test samples. This procedure can prevent the fallacy of cherry-picking in numerical experiments and is described further in Section 5.1.2.

5.1.1. Data Description and Preprocess.

We test the performance of these approximation algorithms in a simulated MoD system with mixed fleets. The ride-pooling simulation follows a batch-to-batch procedure similar to that employed in Alonso-Mora et al. (2017), with a demand forecast module to maximize the expected total value realized by serving travel demand.

Table 1 summarizes our experiment settings. The primary data inputs include the following:

1. Road networks: The road network in Manhattan, NYC is obtained from the OpenStreetMap data. The average traveling time on each road segment is computed using the historical speed data in (Sundt et al. 2021). In Setting 1, both vehicle types can serve any nearby ride requests made in Manhattan. To demonstrate AVs' early deployment in Setting 2, two AV zones are selected in the planning phase (Figure 9(a)). These zones are highly congested areas proposed for pedestrianization and could potentially be closed off to most vehicles besides MoD transit services. Because of regulations for safety concerns, automated MoD buses only operate within these AV zones (Chen et al. 2017b).

2. Supply: The basis set S_B represents ride-hailing vehicles with a fixed capacity of two that provides the standard service.

   - In Setting 1, the augmented set S_A represents a set of locations to which premium vehicles can be repositioned (Figure 6). Full-time drivers provide reservation-based service with these premium vehicles, which have a capacity of three passengers (Ma

Table 1. Parameters in Numerical Experiments

	Augmented set S_A			Basis set S_B					
Setting	Capacity	Location no. $\lvert S_A \rvert$	K	Capacity	Vehicle no. $\lvert S_B \rvert$	Problem statement	Demand-to-supply ratio	Matching interval (min)	Number of sample scenarios N
Setting 1	2	115	60	2	60	Mid-Capacity SRAMF for Example 1 (standard-premium cars)	1.7–2.0	1	10
Setting 2	10	30	5	2	115	High-Capacity SRAMF for Example 2 (mixed autonomy)	35–45	10	50

et al. 2017). Thus, the MoD platform can allocate at most K idling premium vehicles to facilitate freelancing drivers who accommodate ad-hoc demand surges for standard services (Dong et al. 2023).

- In Setting 2, the augmented set S_A represents the initial parking locations from which automated MoD buses are repositioned (Figure 9). Each shuttle bus in S_A has a capacity of up to ten passengers and can operate only within AV zones at the beginning of intervals. If each MoD bus can be repositioned only to a particular subset of locations, we use the generalized setting in Section 4.3, where K_m represents the set of approachable locations for vehicle m. The same approximation ratio holds for this extension.

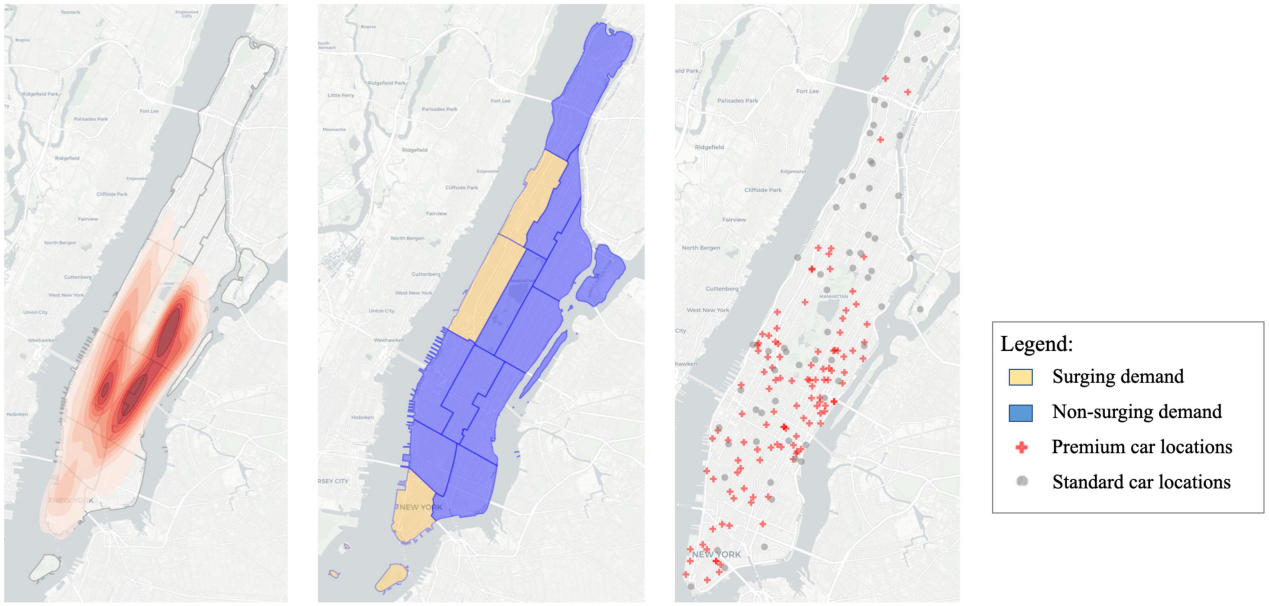
3. Demand: We create a demand forecast model to sample ride requests from the NYC Taxi and Limousine Commission trip data (TLC 2021). The forecast model uses this data set's origin-destination, number of passengers, trip time, and fare information to predict hyperedge values as accurately as possible (Figure 9(b)).

4. Hyperedge values: The value of each hyperedge e is computed by (1). Each trip's pickup time follows the shortest path connecting all ride requests in the hyperedge e, and customers' preference over mixed fleets is randomly generated such that $v_e > 0$.

5. Time intervals: We choose different matching intervals in Setting 1 and Setting 2 in the batch-to-batch implementation. Setting 1 uses a one-minute interval to provide

Figure 6. (Color online) Midcapacity Mixed Autonomy Traffic Experiment in Manhattan, NYC

(a) Demand density (b) Surging demand areas (c) S_A and S_B for Setting 1 in a given scenario



convenient and responsive MoD services; Setting 2 uses a ten-minute interval to permit repositioning between AV zones. Choosing the relatively large matching interval (ten-minute) also illustrates the scalability of approximation algorithms, whereas Qin et al. (2021) showed that the optimal interval might depend on the supply-demand relationship.

5.1.2. Assessment of Algorithms in Mixed-Fleet Simulations. The objective of both settings established in Section 5.1 is to choose a subset of locations for repositioning vehicles S_A to maximize the total expected assignment value. Consequently, the assessment of different benchmarks and proposed approximation algorithms consider three metrics: total computational time (runtime in seconds), the expected assignment value based on predicted demand samples (the objective value of the SRAMF problem defined in (2)), and the average realized assignment value of the proposed locations given new demand samples. In this section we describe these assessment methods and how they are calculated. The numerical experiments conducted in this study generate a fixed number of scenarios, each of which constructs a shareability graph using the process outlined in Appendix B and creates all hyperedges with only positive values.

We measure runtime as the duration in seconds it takes an algorithm to produce the desired output, a subset of augmented supply locations, given the inputs of a hyperedge graph (including associated hyperedge values and the samples of predicted demand). In Benchmark 1, the SRAMF problem is solved to optimality using Gurobi 9.1, a highly optimized MIP solver. The number of variables in Equation (2) equals the product of the number of hyperedges and the sample size, which may increase exponentially in real-world applications. Hence, we set a six-hour computation time limit for solving the SRAMF problem with any method, mainly for the exact solver. Unlike in Benchmark 1, our proposed approximation algorithms focus on solving the SRAMF problem by swapping vehicle locations or adding them sequentially from the set S_A, rather than solving the entire two-stage integer program at once. As a result, a parallel-computing scheme can considerably reduce the total runtime of approximation algorithms by evaluating multiple scenarios concurrently. We report two computation times for these parallelized versions, one for the observed runtime (programmed by Python 3.8 on our server) and another for the hypothetical runtime based on a maximum number of threads. The maximal computing resource (max-thread) runtime limit means that the algorithm can simultaneously evaluate all pairs of candidates in the active set of LSLPR or the dual variables for all hyperedges in MMO. This limit includes additional time for computations in series but excludes the overhead cost of creating processes. The number of used threads for each experiment setting is provided in the footnotes of Tables 2 and 3. Computation times are reported from performance on a server with an 18-core 3.1-GHz processor and 192 GB RAM. In some experiments, the overhead cost of parallelizing the MMO algorithm in Python actually increases the overall computation time, so we report a simply vectorized version. Given the inefficiency of the parallelizing process in Python, we believe times closer to the parallel limit can still be achieved.

The computation times of generating demand forecasts and the corresponding shareability graph are not reported, because this study focuses on reducing the computation time for a given shareability graph (hypergraph) in the SRAMF problem. In practice, the construction processes of these hypergraphs may include more sophisticated demand forecast models (Geng et al. 2019), so their preprocessing times are not considered in the assessment. Approximation algorithms can handle more extensive shareability graphs or require significantly shorter computation time limits. However, measuring their optimality gaps requires comparing objective values of approximation algorithms with that of benchmark methods and downsampling from the original taxicab data. The end of this section appends a separate set of experiments to demonstrate the scalability of approximation algorithms with larger instances.

Additionally, we evaluate the performance of our algorithms by comparing the objective value they achieve to that of the optimal IP solution, commonly known as the optimality gap. Let the objective of the IP solver be OPT and the approximation algorithm's solution be ALG. The optimality gap is measured by $(OPT - ALG)/OPT$ and is reported as a percentage.

Finally, to assess the value of incorporating stochasticity in ride-pooling matching, we compare the average performances of these algorithms by dividing the data set into training and test samples. They are evaluated based on 10 new test samples drawn from the same distribution $F(\xi)$ as the training samples used for initializing the algorithms. Each algorithm and benchmark returns a subset of augmented vehicle locations from the training samples, which are optimally assigned to incoming demand in the next interval. For each test sample, we calculate the optimal matching that can be achieved given the selected augmented supply vehicles. The performance is calculated as a multiplier of the assignment value achieved by Benchmark 1 within each realized sample, which are then averaged across all 10 test samples to give the reported *average test performance*. Note that this multiplier can be greater than one, as the IP solver often produces locations that are optimal for the training samples but may not generalize well to the overall distribution. The IP solver may also reach the computation time limit and find a suboptimal solution. Benchmark 2 was calculated by solving a deterministic second-stage assignment problem (3) with average demand forecasts, reducing it to a much easier-to-solve integer assignment program.

Table 2. Summary of Numerical Results for Mid-Capacity SRAMF

Base, surge demand sample rate	Surge problem	No. of samples	Benchmark 1	Benchmark 2		LSLPR			MMO		
			IP runtime (s)	Average test performance	Runtime (s) (36-thread)	Max-thread runtime (s)	Optimal gap (%)	Average test performance	Runtime (s) (vectorized)	Max-thread runtime (s)	Optimal gap (%)	Average test performance
(0.4, 0.6)	0.3	10	1,109	1.00	1,818	19	**0.1**	1.15	41.1	**0.26**	3.1	1.14
		20	4,161	0.98	2,682	28	**0.6**	1.14	52.8	**0.35**	3.1	1.13
	0.5	10	1,117	1.00	1,178	12	**0.2**	1.19	43.8	**0.26**	2.7	1.14
		20	4,213	0.99	—	273	**0.0**	1.18	53.5	**0.34**	4.5	1.12
(0.5, 0.75)	0.3	10	1,160	1.00	3,742	39	**0.1**	1.21	48.7	**0.28**	2.6	1.18
		20	4,202	0.98	3,893	41	**0.2**	1.19	56.1	**0.41**	5.2	1.14
	0.5	10	1,177	1.01	3,090	33	**0.1**	1.25	43.0	**0.29**	8.5	1.16
		20	4,214	0.98	6,365	67	**0.0**	1.22	58.3	**0.42**	10.4	1.12

Notes. The maximum computation time is three hours. —, hitting the maximum runtime. Bold entries are the best results.

5.2. Numerical Results for Mid-Capacity SRAMF

Setting 1 uses large fleets of premium and standard vehicles to provide ride-pooling services in tandem. The demand density, surging demand areas, and two sets S_A, S_B are shown in Figure 6. We generate data for surging demand as follows: (1) divide Manhattan, NYC, into 12 regions (using NYC Community Districts (Data 2022)) and (2) create a probability matrix of all regions to chart the occurrence of surging demand. For concision, we choose three regions with high probabilities of surges (Figure 6(b)); the remaining regions have a low probability of surging demand. These regions were chosen to be outside of areas with high demand at the time of prediction to test the quality of service with highly fluctuating demand distributions. The surging magnitude is defined by the multiplier of sampling rates of surging vs. the nonsurging cases from the actual trips from the NYC taxi data set. (3) Generate i.i.d. demand profile and build shareability graphs for each scenario. (4) Run all algorithms on the same sample set, using benchmark models, and evaluate numerical results.

Because the shareability graph is constructed for each scenario, the computed optimal routes in a sample scenario are shown in Figure 7. As can be seen, premium vehicles selected from the augmented set S_A pick up mainly customers in the surging demand areas. Because the hyperedge values of these rides are likely to have a surge multiplier, the platform tends to reposition more idling premium vehicles and switch them to serve standard requests. The computational results for Setting 1 are summarized in Table 2.

5.2.1. Computation Times. At small sample sizes, the IP's runtime is fairly comparable to that of the LSLPR approximation algorithm. Gurobi is a powerful and heavily optimized MIP solver, so this result is unsurprising. MMO and LSLPR have a clear advantage at larger sample sizes because they can use parallel computation resources. MMO is also the only algorithm to complete within the time limit for this setting, although many LSLPR runs can also reach this threshold with enough parallel threads.

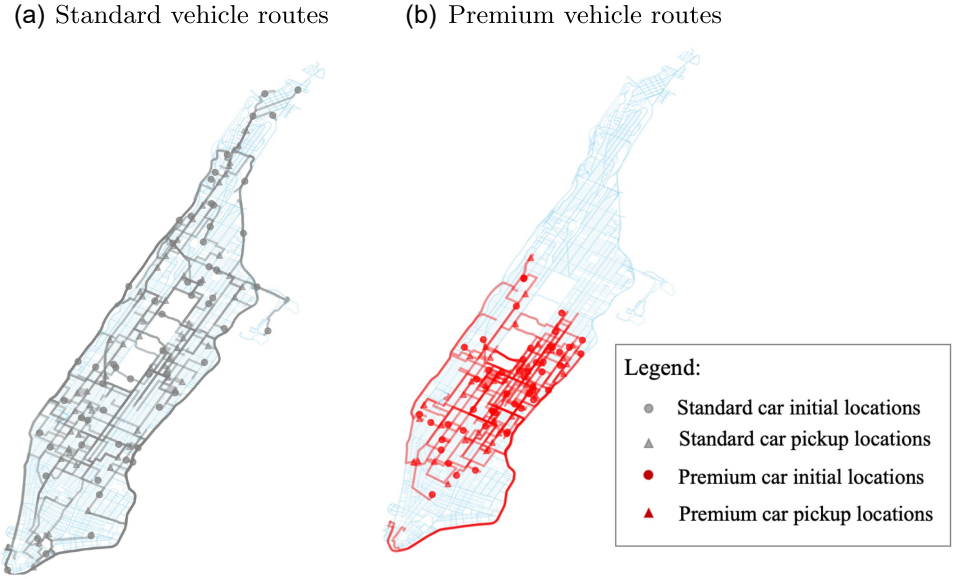
The runtime of LSLPR varies widely as the swap order greatly affects the runtime. When the value ϵ in the stopping criterion is small in cases analyzed by large-scale shareability graph, long runtimes for approximation algorithms are occasionally observed. This is primarily because the algorithm must evaluate many swaps to find one that improves the overall objective value. If the algorithm randomly initializes with a competitive solution of S_R, finding another swap to improve the objective value becomes increasingly difficult. This can lead to the algorithm's evaluating a combinatorial number of swaps per iteration, but the quality of such an approximation attains near optimality. One advantage of LSLPR is that it can be stopped early and still return valid assignments

Table 3. Summary of Numerical Results for High-Capacity SRAMF

			LSLPR			MMO		
Demand-supply ratio	S_A, S_B	Benchmark 1 IP runtime (s)	LSLPR runtime (8-thread) (s)	LSLPR runtime (max-thread) (s)	Optimal gap	MMO (8-thread) (s)	MMO (max-thread) (s)	Optimal gap
1.78	10,115	1,177	384	20	**0.2%**	**38**	**16**	0.4%
1.78	15,115	1,332	383	**19**	0.9%	**51**	**19**	**0.8%**
1.78	20,115	1,470	337	23	**0.7%**	**58**	**15**	0.8%
1.78	25,115	1,354	357	**24**	0.9%	**72**	26	**0.8%**
1.78	30,115	1,464	548	31	**0.5%**	**82**	**15**	0.9%
1.78	35,115	1,448	599	27	**0.4%**	**97**	**17**	1.0%
1.83	10,115	1,425	460	29	0.3%	**48**	**15**	**0.1%**
1.83	15,115	1,516	509	63	0.1%	**60**	**17**	**0.0%**
1.83	20,115	1,730	449	33	0.5%	**77**	**19**	**0.3%**
1.83	25,115	1,817	483	35	**0.3%**	**91**	**22**	0.7%
1.83	30,115	2,027	566	31	**0.7%**	**104**	**21**	0.8%
1.83	35,115	2,373	565	44	**0.5%**	**137**	**31**	1.2%
1.87	10,115	1,180	353	**26**	0.7%	**87**	65	**0.3%**
1.87	15,115	1,350	398	**12**	0.5%	**66**	34	**0.1%**
1.87	20,115	1,509	455	55	1.0%	**78**	**38**	**0.1%**
1.87	25,115	1,650	488	**18**	1.0%	**128**	70	**0.3%**
1.87	30,115	1,690	536	35	1.0%	**105**	**32**	**0.3%**
1.87	35,115	1,795	535	77	1.3%	**141**	**42**	**0.3%**
1.93	10,115	4,468	1,011	196	0.4%	**121**	**63**	**0.2%**
1.93	15,115	6,411	1,036	**58**	0.5%	**250**	140	**0.4%**
1.93	20,115	11,243	1,237	204	**0.2%**	**224**	**72**	0.3%
1.93	25,115	11,820	1,318	**28**	0.4%	**390**	167	**0.2%**
1.93	30,115	5,432	1,453	99	0.8%	**397**	**86**	**0.6%**
1.93	35,115	7,038	1,830	**64**	0.6%	**516**	133	**0.4%**
2.02	10,115	4,348	998	**50**	**0.6%**	**226**	182	1.0%
2.02	15,115	5,843	1,792	157	**0.2%**	**175**	**93**	1.0%
2.02	20,115	9,802	2,177	**210**	**0.2%**	**442**	214	1.0%
2.02	25,115	16,787	3,508	300	**0.5%**	**609**	**201**	0.9%
2.02	30,115	22,141	4,143	**198**	0.6%	**1,234**	477	**0.4%**
2.02	35,115	20,920	6,327	**161**	**0.4%**	**1,362**	487	0.5%

Notes. The demand-supply ratio is the average ride requests over the total number of vehicles ($K + |S_B|$); $K = 5$. The max-thread runtimes assume enough threads to evaluate all potential swaps or drivers at once. The total number of variables in Benchmark 1 (IP) ranges from 3×10^6 to 1.2×10^7. Bold entries are the best results.

Figure 7. (Color online) Optimal Trip Assignment and Routes in the Midcapacity Scenario



without searching all swaps, which other algorithms and benchmarks cannot do. We verify this hypothesis by the following observation: With $\epsilon = 0.1$, Table 2 shows the very small optimality gaps this algorithm can achieve. LSLPR becomes time-competitive in runtime for most instances (Figure 8) when ϵ increases to 0.2.

5.2.2. Optimality Gaps. Table 2 shows that the actual optimality gaps throughout the numerical experiments are smaller than the theoretical bounds. The optimality gaps of LSLPR (≤1%) are significantly smaller than those of MMO (3%–10%), which matches the theoretical analysis.

Regarding performance on the test samples, LSLPR and MMO vehicle selections consistently achieve 12%–25% higher objective values than those of Benchmark 1 (IP solver). This advantage is always slightly worse when more samples are used initially, demonstrating that the benchmark is optimizing heavily to the specific training samples. This result does not generalize to demand distributions as well as LSLPR or MMO. Although Benchmark 1 might outperform our algorithms on some test samples and obtain a larger objective value with enough initial samples, the IP solver scales poorly in regard to the average runtime (Table 5). Additionally, the deterministic benchmark shows that disregarding demand uncertainty entirely can cause a 15%–22% loss of objective value in the worst case when compared with values that LSLPR and MMO achieved. This is a significant difference considering that the future MoD market is a billion-dollar industry. Therefore, it is valuable to implement SRAMF algorithms to proactively reposition vehicles as in ride-hailing literature (Qin, Zhu, and Ye 2021) instead of solving the deterministic problem.

5.3. Sensitivity Analysis

To evaluate the impacts of this parameter on the computational efficiency, we test the computation times and optimality gaps with varying $K \in [10, 80]$. The results in Figure 8 show that the computation times of approximation algorithms increase significantly with K, which is a shortcoming of any local-search-based algorithm. The IP solver is relatively unaffected by the change of parameters. However, the optimality gap of these approximation algorithms drops to nearly zero with the increasing budget, as the increased budget has diminishing marginal value to the ride-pooling assignment (i.e., the market is saturated already). Therefore, the platform can select a

Figure 8. (Color online) Impact of K on Computation Time and Optimality Gap in the Midcapacity Scenario

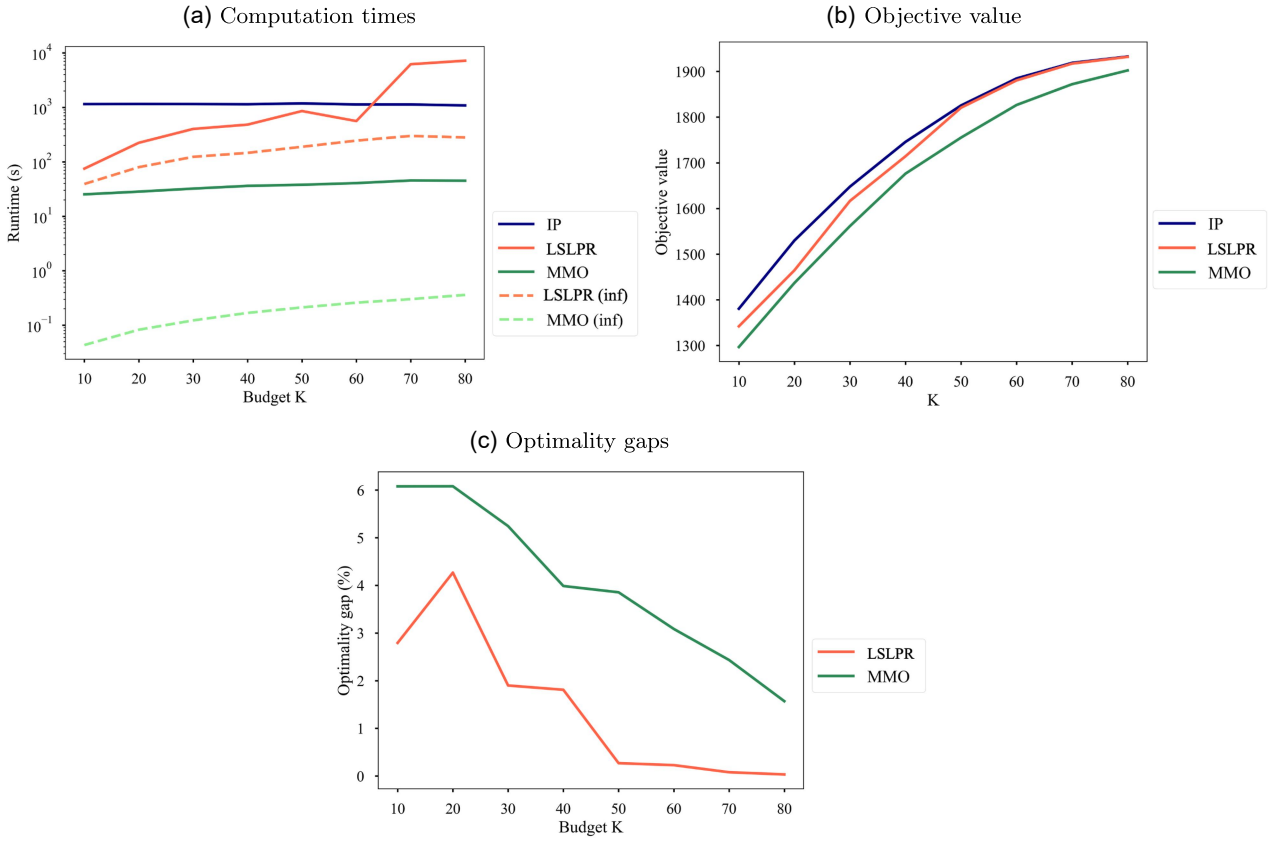


Table 4. Impact of Vehicle Capacity on Computation Time and Optimality Gap

Number of AV locations	AV capacity C_{AV}	Number of hyperedges	Runtime of IP (second)	Optimality gap of LSLPR (%)	Optimality gap of MMO (%)
35	2	8,121	220	**0.62**	1.12
35	4	11,396	288	**0.80**	0.88
35	6	12,048	299	0.99	**0.43**
35	8	12,068	298	1.00	**0.03**
35	10	12,070	301	1.00	**0.02**

reasonably small budget for approximation algorithms to have clear advantages over the exact solver.

5.4. Numerical Results for High-Capacity SRAMF

In Setting 2, the proposed algorithm computes near-optimal solutions for the mixed-autonomy fleet, including repositioning automated MoD buses (AVs) among locations S_A (Figure 9) and determining their pickup routes for demand samples.

The performance of the proposed approximation algorithm is evaluated under different supply and demand distributions. The system is tested in both a relatively balanced demand scenario as well as a massive undersupply scenario, with mean numbers of demand across scenarios ranging from 250 to 4,000, respectively, which are considerably larger in a *stochastic* setting. Figure 10 shows that algorithms allocate four AVs to the high-demand-for-AV zone and one AV to the low-demand-for-AV zone for such demand forecasts. Notice that these decisions are complementary to CVs' trip assignment decisions, as the latter fleets are still the primary MoD service providers.

5.4.1. Computation Times. The reported computation time includes solving for the near-optimal vehicle selection and exact assignment in each scenario. This comparison excludes the runtime required to generate hypergraphs to emphasize the performance of algorithms in solving the SRAMF problem. Recall that the size of the augmented set $|S_A|$ and the number of hyperedges (the number of decision variables in each scenario) determine the size of the shareability graph. Table 3 shows how the total runtime grows with the increasing size of the hypergraph. The computation time of LSLPR and MMO algorithms are shown in Table 3.

The largest runtime per iteration is reported in Table 3, where the number of hyperedges is the maximal number across all scenarios. Our results show the following. (a) The max-thread MMO setting obtains near-optimal solutions to SRAMF with the smallest runtimes because it evaluates all potential vehicles in $S_A \setminus S_R$ in parallel. (b) The performance of LSLPR is worse than MMO for high-capacity SRAMF, which matches our approximation ratio analyses where the number of potential swaps grows exponentially. Nevertheless, its computation time will always improve when additional resources are available.

5.4.2. Optimality Gaps. Table 3 shows that optimality gaps of both algorithms grow slightly with the size of shareability graphs, but the overall performance of the proposed approximation algorithms is satisfactory for various supply-demand ratios. The optimality gaps are below 2% throughout tested instances, confirming that approximation ratios derived for the worst-case scenario, $1/p^2$ or $(e-1)/(2e+o(1))p\ln p$, are loose with the real-world trip data. In other words, the performance degradation of these approximation algorithms is negligible when implementing them in shared mobility systems.

5.4.3. Sensitivity Analysis. Three sensitivity analyses for the high-capacity SRAMF problem involve (a) distribution of hyperedge values, (b) vehicle number and capacity, and (c) sample size. They test how the performance of these approximation algorithms is affected based on changes in input data and model assumptions. We discuss them individually in this section.

5.4.3.1. Hyperedge Value Distribution. The first set of sensitivity analyses aims to check algorithms' performance

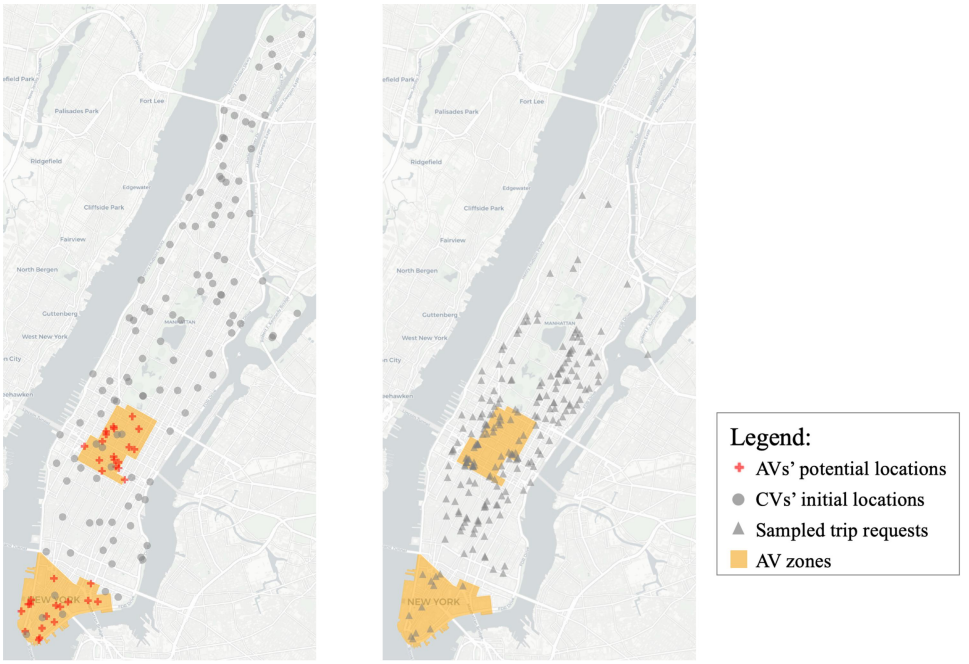
Table 5. Impact of Sample Size on Computation Time and Optimality Gap

Number of samples	Number of AV locations	AV capacity	Average number of hyperedges	Runtime of IP (s)	Runtime of LSLPR (s)	Optimality gap (%)
10	20	5	3,100	14	2	3.23
25	20	5	3,100	104	5	1.63
50	20	5	3,100	445	12	2.03
75	20	5	3,100	907	15	2.01
100	20	5	3,100	2,088	25	0.91
150	20	5	3,100	4,076	38	3.15
200	20	5	3,100	7,485	109	1.01

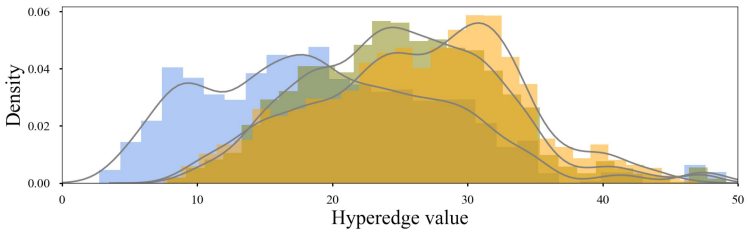
Figure 9. (Color online) High-Capacity Mixed Autonomy Traffic Experiment in Manhattan, NYC

(a) CVs' and AVs' initial locations in Setting 2

(b) Request pickup locations in a sampled experiment



(c) Sampled hyperedge value distributions in shareability graphs



degrades with different supply and demand distributions. By replacing the empirical hyperedge values with randomly generated hyperedge values, this analysis examines the robustness of these algorithms. Figure 11 shows the runtime and optimality gaps with uniformly generated hyperedge values. Figure 9(c) represents that customers' level of trust in the AV technology dominates the hyperedge value such that $v_e = \sum_{j\in t} u_j + \sum_{j\in t} \tilde{u}_{ij} - c(i,t) \approx \sum_{j\in t} \tilde{u}_{ij}$ for each $e \in E(\xi)$ and $\tilde{u}_{ij}$ follows a uniform distribution.

The runtime of random hyperedge values is smaller than those of real-world data, and the optimality gaps stay low across most instances. This is mainly because the empirical hyperedge values are more concentrated around specific values (i.e., the average trip length). Hence it is more difficult to reposition vehicles from the augmented set. In this case, the local search–based algorithms outperforms other algorithms with uniformly distributed values.

5.4.3.2. Vehicle Capacity. In Setting 2's numerical experiments, automated MoD buses (AVs) provide mixed autonomy mass transport whose vehicle capacity is up to 10 passengers. CVs have a fixed capacity of three passengers. Recall that p bounds the vehicle capacity. Table 4 shows how the vehicle capacity affects the approximation ratios. A surprising observation is that the vehicle capacity is not the bottleneck of approximation algorithms' performance throughout the experiments. In contrast, the IP benchmark's computation time increases significantly vehicle capacities. This is because the number of hyperedges plateaus above a certain capacity due to the process of constructing shareability graphs outlined in Section 5.1. For a high-capacity trip to

Figure 10. (Color online) Optimal Trip Assignment and Routes in Mixed Autonomy, High-Capacity SRAMF

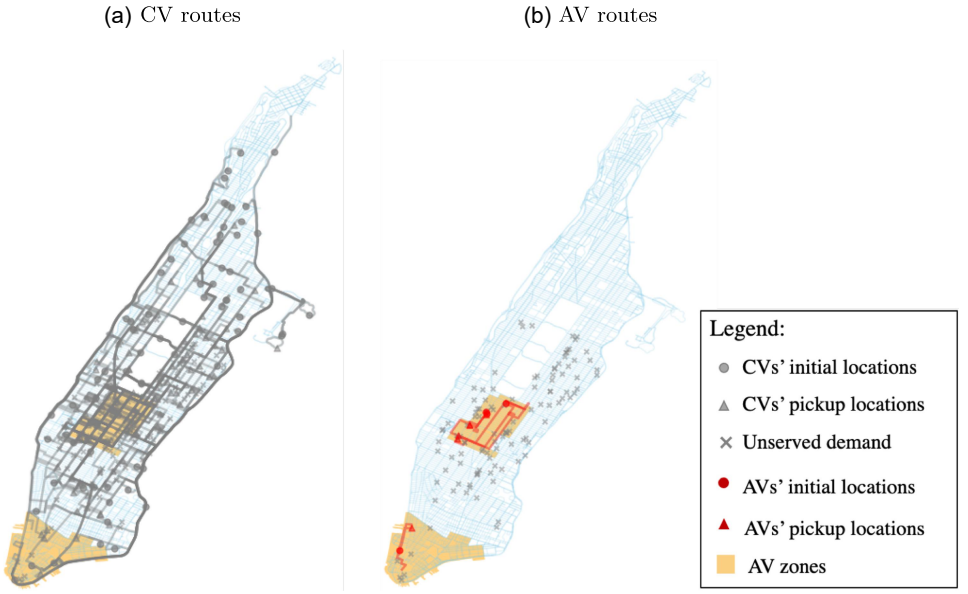
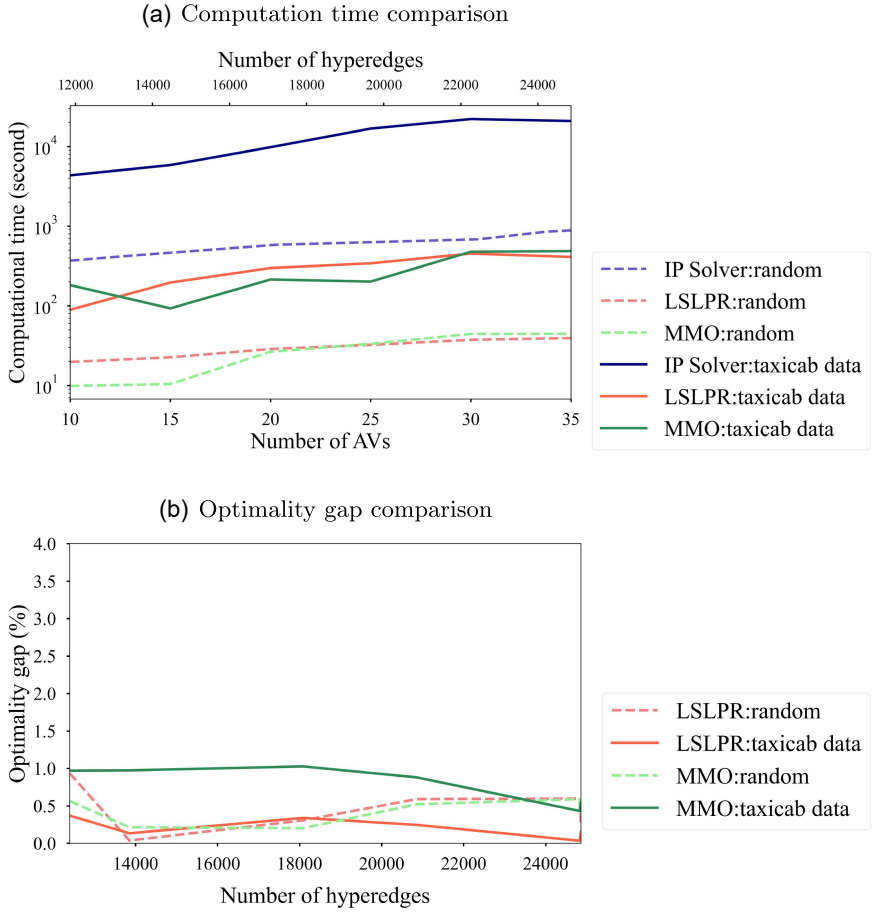


Figure 11. (Color online) Impact of Input Distribution on Computation Time and Optimality Gap



exist, that is, the corresponding hyperedge value is positive, all subset trips must also exist. This requirement leads to a combinatorially decreasing number of hyperedges with large trip sizes unless an even larger set of compatible trips exist. Because the density of ride requests in the AV zones does not satisfy the existence conditions for compatible trips, the optimality gaps of both approximation algorithms are not evidently affected by the AV capacity.

5.4.3.3. Sample Size. The SAA method guaranteeing a uniform convergence to the optimal value does not directly reveal how the sample size affects the total computation time of solving the SRAMF problem. Table 5 summarizes the computational results of algorithms compared with the same instances, which reports runtimes with finite and maximal computational resources (i.e., number of threads for parallel computing).

The approximation algorithms address demand uncertainties in MoD platforms while controlling the computational time to increase linearly with the sample size. Table 5 demonstrates that optimality gaps are small for all instances investigated. In light of the important nature of stochastic demand and the movement of basis vehicles, our numerical results suggest that MoD platforms should employ more computational resources to facilitate approximation algorithms with large sample sizes to improve the platform's average profit and quality of service.

6. Conclusion

SRAMF uses a two-stage stochastic integer program to calculate joint vehicle repositioning and assignment for large-scale MoD systems with mixed fleets. This research introduces two approximation algorithms, LSLPR and MMO, which leverage the structure of shareability graphs to assess potential matchings with increased supply. These algorithms efficiently generate near-optimal solutions for maximizing the expected total value of ride-pooling assignments in a relatively short time. The main theoretical results provide provable guarantees for their worst-case performance, as validated by extensive numerical experiments involving midcapacity and high-capacity vehicles. Our results illustrate the significant benefits of integrating stochastic programming modules into the MoD vehicle dispatching processes to improve system profitability and throughput.

To close this paper, we point out several promising future research avenues to address the following limitations. First, alternative pickups and dropoffs in trip planning are not permitted, i.e., the total number of ride requests per hyperedge is less or equal to vehicle capacity. Second, as the SRAMF problem only considers a two-stage uncertainty structure, it is meaningful to extend this framework to a multistage setting with time-varying demand forecasts and vehicle repositioning decisions that adapt to revealing scenarios. Constructing hypergraphs with multistage demand forecasts will be a major computational bottleneck. Hence, new representations for potential matchings must be developed to address computational issues. Third, penalties related to balking trips or carryover supply are not directly considered in the current setting, whereas they can be incorporated into the hyperedge value in Section 3.1. Finally, the current trip assignment does not consider cancellation and reassignment after dispatching vehicles to passengers. Considering these factors in practice may improve the stability of ride-pooling algorithms.

Appendix A. Summary of Notation

Table A.1. Summary of Notation and Acronyms

Notation	Description
S_A, S_B	Augmented set and basis set of vehicles
ξ	Randomly generated scenario
$\ell \in [N]$	Index for sampled scenarios and the total number of samples
$D(\xi)$	Set of demand in scenario ξ
$E(\xi)$	Set of hyperedges in scenario ξ
G	Shareability graph, a hypergraph consists of supply and demand vertices and hyperedges
e	Each hyperedge $e = \{i, J\}_{i \in S, J \subseteq D}$ is a potential trip where vehicle i serves requests J
u_j	The expected profit of request j
$\tilde{u}_{ij}$	Utility gained from matching request j with preferred vehicle type i
$c(i, t)$	Travel cost for vehicle i to serve trip t
α	Approximation ratio
n	Total number of vehicles such that $\lvert S_A \rvert = n_A$ and $\lvert S_B \rvert = n_B$
K	Maximum number of vehicles allowed from the augmented set
C_i	Capacity of vehicle $i \in S_A \cup S_B$
w_i	Number of passengers in request j
p	Maximum capacity of hyperedge, $p = \max_{i \in S_A \cup S_B} \{1 + C_i\}$
j	Index for travel demand $j \in D(\xi)$

Table A.1. (Continued)

Notation	Description
w_j	Size of travel demand $j \in D(\xi)$
$E_{i,\xi}$	Set of hyperedges contains vertex $i \in S_B \cup S_R$ in scenario ξ
t	Trip is a set of demand following the shortest pickup-and-then-dropoff order
v_e	Value of hyperedge $e \in E(\xi)$
$nb(e)$	Neighboring hyperedges $e' \in E(\xi)$ intersecting with e
x_e	Decision variable for hyperedge e, $x_e \in \{0,1\}$
$\overline{x}_e$	Decision variable for fractional assignment, $x_e \in [0,1]$
y_i	Decision variable for vehicle $i \in [S_A]$, $y_i \in \{0,1\}$
$v^*(\cdot)$	Optimal value of the exact GAP
$Q(y,\xi)$	Optimal value of the assignment in scenario ξ
$v_{\max}$	Maximal hyperedge value for all $e \in E$
$v_{\min}$	Minimal hyperedge value for all $e \in E$ such that $v_e > 0$
$\mathcal{I}$	Independent set as a union of hyperedges satisfying the set-packing constraint
S_O	Optimal choice of vehicles for SRAMF $S_O \subset S_A$
S_R	Choice of vehicles from the algorithm $S_R \subset S_A$
$\mathcal{L}$	The bijection between S_R and S_O
$\hat{v}(\cdot)$	The objective value of fractional assignment
$\boldsymbol{z}$	Optimal LP solutions to $\hat{v}(S_O)$
F_i	Hyperedges intersect with vehicle i
H_d	Hyperedges intersect with demand d
$\perp$	Dummy hyperedge in LSLPR
$\Delta_d(e,f)$	Decomposition mapping between hyperedge e and f
$U_{i_1 i_2}$	Marginal value function with $i_1 \in S_R$ and $i_2 \in S_O$
$\hat{v}_{ON}$	Objective value of the online algorithm
$u_{g,\xi}$	Dual variable in the MMO algorithm for $g \in e$ and scenario ξ
Γ_e	$\Gamma_e = \sum_g u_{g,\xi}$ as the left side of dual constraints
$\boldsymbol{c}_e$	Row of cost coefficient in the dual covering problem with entry $c_e(g,\xi)$
M	The augmented set is partitioned into M subsets
ϵ	Error tolerance (for stopping criteria)
δ	Error tolerance (for sample average approximation) or constant in MMO update subroutine
OPT	Optimal value of the SRAMF problem
ALG	Objective value of solving SRAMF by approximation algorithms
Acronym	
CV/AV	Conventional/automated vehicle
GAP	General assignment problem
LP	Linear program
IP	Integer program
LSLPR	Local-search linear-program-relaxation algorithm
MMO	Max-min online algorithm
SAA	Sample average approximation
SRAMF	Stochastic ride-pooling assignment with mixed fleets
VRP	Vehicle routing problem
DVRP	Dynamic vehicle routing problem

Appendix B. Performance Analysis of Construction of Shareability Graphs

The main idea of recent ride-pooling assignment papers (Santi et al. 2014; Alonso-Mora et al. 2017; Simonetto, Monteil, and Gambella 2019) is to separate the problem into two parts: (1) constructing the shareability graph and compatible requests and vehicles and (2) optimally assigning those trips to vehicles by solving GAP. This paper primarily focuses on algorithms and approximation bounds for the stochastic extension to the second part. Still, we acknowledge the importance and difficulty of the first task and describe them in detail below for completeness.

B.1. Procedure for Constructing Shareability Graphs

$D(\xi)$ is a set of all ride requests revealed in scenario ξ, and this section omits ξ when there is no confusion because the hyperedges for scenarios are generated separately. We consider many parameters to be given by the customer or externally dictated to the platform (based on desired service parameters). These include, for each customer j, the maximum waiting time, ω_j, and allowable delay, r_j.

- *(Constraint I)* Travel time from vehicle location to pickup of customer j in order must be less than ω_j.
- *(Constraint II)* Travel time from origin to destination of customer j in order must be less than r_j.

Additionally, as defined in the setting, the hyperedge weight consists of three parts: value of ride requests $\sum_j u_j$, preference of the vehicle type $\tilde{u}_{sj}$, and travel cost of delivering all ride requests in a single trip. We take a three-request clique (j_1, j_2, j_3) as an example. Let $t_k = \{O_{j_1}, O_{j_2}, \ldots, D_{j_2}, D_{j_3}\}$ be a specific ordered sequence of origins and destinations and $SP(t_k)$ be the shortest path route connecting them. Let

$T_e = \cup_k t_k$. This is slightly less demanding than finding all feasible Hamiltonian paths if we enforce that, in all trips, the origins must be picked up before any destination is visited. Let $c(s,e) = \min_{t_k \in T_e} v(t_k)$, where $c(t_k)$ is the cost of serving all requests following the shortest-path $SP(t_k)$. We define a set function $f(e)$ that takes a hyperedge consisting of a vehicle s and a potential combination of trips, T_e, as follows:

$$f(e) = \begin{cases} 0 & \text{if } \forall t_k \in T_e,\ SP(t_k) \text{ violates constraints I and II} \\ c(s,e) & \text{otherwise.} \end{cases}$$

The bottleneck of computation time is still finding vehicle routes that satisfy the given constraints by solving a constrained VRP problem, which is NP-hard. Therefore, all heuristic methods can only minimize this bottleneck as much as possible by reducing the number of combinations to check at each step. For example, Ke et al. (2021) suggested a reformulation for finding $c(s, e)$ to avoid enumerating all possible paths.

We combine multiple heuristic methods in literature to construct the shareability graph. First, we must identify the valid single-customer trips for a given vehicle s. Let D_s be the demand that can be served by vehicle s in a single trip within the allowable pickup time. We may further reduce the number of trips by planning on a spatiotemporal graph and examining compatible trip cliques. By testing trips in order of increasing size and only considering a trip if all subsets of trips (where one request is removed from the trip) are feasible, we reduce the number of candidate trips by orders of magnitude. This heuristic generates the shareability graph in Figure 2 in which a set of requests is tested for trip compatibility only if every subset of that set of requests is also compatible.

Lemma B1 (Alonso-Mora et al. 2017). *A trip associated with the hyperedge e is feasible for vehicle s only if, for all $j \sim e, j \in D_s$, hyperedges $e' = e \backslash \{j\}$ are feasible.*

The heuristic reduces the candidate hyperedge sets by leveraging the topological relationship between matchable trips of size k and $k + 1$ (Figure B.1), without eliminating potentially feasible trips. The hypergraph can then be constructed in order of increasing capacity to minimize the number of request sets tested. Additionally, we adopt the following rules to further reduce the number of candidate trips:

1. Because only hyperedges with nonnegative edge weights are of interest, we remove all the trips from the candidate set subject to $f(e) <= 0$.
2. If a vehicle v is not feasible for trip t_k at time τ, it will not be feasible for t_k at any time $\tau' > \tau$ (Liu and Samaranayake 2020).

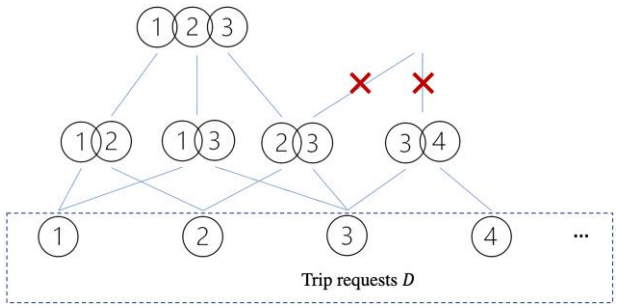
Let $C_k(D)$ be the set of combinations of size k of the elements of the set D. This process is summarized as follows.

Algorithm B.1 (Construction of Shareability Graph)

Data: Vehicle locations and requests (request time, pickup, drop-off, preferred vehicle type, acceptable delay)

Result: Set of hyperedges, E, each containing a vehicle, s, and a set of compatible requests for that vehicle to serve in one trip. Hyperedge values are v_e for all $e \in E$.

Figure B.1. (Color online) Topological Relationship Between Cliques of Matchable Requests



Note. In this example, (2,3,4) is not a valid combination of requests because the (2,4) combination was not valid.

Initialize $E = \emptyset$
for $s \in S_A \cup S_B$ **do**
  Identify candidate passengers
  $D_s^1 \leftarrow \{e \in D | f((s,e)) > 0\}$
  Add hyperedges of size one
  $E_k \leftarrow \cup_{e \in D_s^1}(s,e)$
  for $k = 2, \ldots, c$ **do**
    for *Demand set* $d \in C_k(D_s^{k-1})$ **do**
      Add trips of size k if all subsets exist and value greater than 0
      if $(s,e') \in E_{k-1}\ \forall i \in C_{k-1}(d)$ *and* $f((s,e')) > 0$ **then**
        $E_k \leftarrow E_k \cup (s,d)$
        $D_s^k \leftarrow D_s^k \cup d$
$E = \cup_{k=1}^{p-1} E_k$
Return hyperedges E and their values v_e

B.2. Performance and Complexity Analysis of the Hypergraph Construction Procedure

B.2.1. Optimality Analysis. The two-step ride-pooling assignment that first constructs the hypergraph and then solves GAP obtains the exact solution of the joint VRP and enjoys the computational advantage for large fleets. Because this work focuses on stochastic assignment, the optimality analysis does not consider the errors of computing hyperedge values. The following results from Alonso-Mora et al. (2017) provide positive guarantees for returning a feasible set of hyperedges in the shareability graph: without enumerating all trip combinations:

1. The optimal value v^* from solving GAP on the shareability graph obtains the optimal value for ride-pooling for an arbitrary batch of supply and demand.
2. The construction of the shareability graph is anytime optimal, that is, given additional computational resources, the set of hyperedges is only expanded to allow for improved matching.

The second property guarantees using a capacity bound, the threshold of which is derived later, as an early stopping criterion in generating the hypergraph will still provide satisfactory results. Solving GAP on this reduced shareability graph can guarantee anytime optimality such that the output is near-optimal for the original problem with high probability.

B.2.2. Computational Complexity Analysis. We consider a fixed sample of demand D and vehicles S in this section as

the hyperedges of each scenario can be generated in parallel. The realized demand has size $|D| = d$.

Lemma B.2. *In the worst case, where all demand is compatible and can be served by all vehicles, the runtime is* $O(|S|d^{p-1})$.

Although in the worst case, runtime is large, this scenario only arises when all trips are compatible with ride-pooling, which is unlikely in practice. Therefore, we consider the Erdos-Renyi model in which an arbitrary pair of demands is matchable (i.e., satisfy the conditions above) with probability q. Empirical studies showed that q was often a small number (<0.1) over a large area (Ke et al. 2021).

Lemma B.3 (Bollobás and Erdös 1976). *The expected number of cliques of size k is* $\binom{d}{k} q^{\binom{k}{2}}$.

For example, with $d = 1{,}000$ and $q = 0.1$, the expected number of cliques of size 3 (each vehicle delivers at most two requests in a single trip) is 500. Often, we observe the size of complete cliques of compatible trips to be less than 10, our maximum tested capacity and the total number of hyperedges is manageable.

Lemma B.4 (Matula 1976). *As* $d \to \infty$, *the maximal clique size* ρ *takes on one of at most two values around* $(2\log d)/(\log 1/q)$ *with probability tending to one, that is, with* $b = 1/q$, $\lfloor 2\log_b d \rfloor < \rho < \lceil 2\log_b d \rceil$.

Therefore, we only need to consider hyperedges with size less than $p^* = \min\{p, \rho + 1\}$ (i.e., the height of the cliques' graph in Figure B.1). We have the following theorem for the runtime of constructing shareability graphs.

Theorem B.1. *In the average case that the demand and supply profiles satisfy the random geometric graph conditions, the runtime is* $O(|S|d^{p^*-1})$.

Proof. The expected number of hyperedges connected to vehicle s, $E_{s,\max}$, is bounded by

$$\begin{aligned} E_{s,\max} &= \binom{d}{1}1^2 + \binom{d}{2}2^2 q + \cdots + \binom{d}{(p^*-1)}(p^*-1)^2 q^{(p^*-2)} \\ &\le ed + \left(\frac{ed}{2}\right)^2 2^2 q + \cdots + \left(\frac{ed}{p^*-1}\right)^{p^*-1}(p^*-1)^2 q^{(p^*-2)} \\ &= ed + \frac{1}{q}\left[\left(\frac{eqd}{2}\right)^2 2^2 + \cdots + \left(\frac{eqd}{p^*-1}\right)^{p^*-1}(p^*-1)^2\right] = O(d^{p^*-1}), \end{aligned}$$

where e is the Euler's number. □

Appendix C. Supplementary Results for Approximation Algorithms

C.1. Proof for Sample Average Approximation in SRAMF

Proof. We denote the optimal value of the SRAMF problem (2) as v^* and the optimal value for the objective from Algorithm 2 as $\hat{v}(S_O)$. Let δ be the upper bound of the optimality gap $v^* - \hat{v}(S_O)$. We assume that $\mathbb{E}[Q(y,\xi)] = \Omega(m^{-2})$ for any ξ where m is a given constant. The main task is to show that $\mathbb{E}[\hat{v}(S_O)] = \Omega(m^2)$ with the sample size $N = m^4/\delta^2$ and $Pr(\hat{v}(S_O) \notin [(1-\delta)v^*, (1+\delta)v^*]) \le \exp\left(-(\delta^2/2)\mathbb{E}[\hat{v}(S_O)]\right)$.

Let $D(\xi) < D$ for all ξ. For any selection of vehicles in S_A denoted by $y \in Y$, $\mathbb{E}[Q(y,\xi)^2] < \infty$, because we can choose K vertices in S_A with maximum number of D edges. The upper bound of hyperedge value v_{ij} is $v_{\max}$. Thus, we have $\mathbb{E}[Q(y,\xi)^2] < K^2|v_{\max}|^2 D^2 < \infty$. Without loss of generality, we draw the following observations from the standard stochastic programming literature (Pagnoncelli, Ahmed, and Shapiro 2009):

1. The objective value $\hat{v}(S_O) \to v^*$ as $N \to \infty$;
2. The expected value $\mathbb{E}[\hat{v}(S_O)] \ge v^*$.

Since N samples are i.i.d., we can use the Chernoff bound on the measure:

$$Pr(\hat{v}(S_O) \notin [(1-\delta)v^*, (1+\delta)v^*]) \le \exp\left(-\frac{\delta^2}{2}\mathbb{E}[\hat{v}(S_O)]\right).$$

Setting $N = m^4/\delta^2$ and using the assumption that $\mathbb{E}[Q(y,\xi)] = \Omega(m^{-2})$, by Jensen's inequality, we have

$$\delta^2 \mathbb{E}[\hat{v}(S_O)] \ge \delta^2 N \cdot E[Q(y,\xi)],$$

that is, $\delta^2 \cdot \mathbb{E}[\hat{v}(S_O)] = \Omega(m^2)$. We have

$$Pr(\hat{v}(S_O) \notin [(1-\delta)v^*, (1+\delta)v^*]) \le \exp(-\Omega(m^2)),$$

which achieves the second task as

$$\begin{aligned} &Pr\left(\left(\frac{1}{2}-\epsilon\right)\hat{v}(S_O) < \left(\frac{1}{2}-\epsilon\right)(1-\delta)v^*\right) \\ &\quad + Pr\left(\left(\frac{1}{2}-\epsilon\right)\hat{v}(S_O) > \left(\frac{1}{2}-\epsilon\right)(1+\delta)v^*\right) \\ &\le Pr\left(\hat{v}(S_R) < \left(\frac{1}{2}-\epsilon\right)(1-\delta)v^*\right) \\ &\quad + Pr\left(\hat{v}(S_R) > \left(\frac{1}{2}-\epsilon\right)(1+\delta)v^*\right) \\ &\le \exp(-\Omega(m^2)). \end{aligned}$$

The first inequality is because $(1/p^2)\hat{v}(S_O) \le \hat{v}(S_R) \le \hat{v}(S_O)$. This concludes the approximation ratio for LSLPR algorithm for the stochastic counterpart of the ride-pooling problem. □

C.2. Proof for Lemma 1

We use a network flow formulation to prove the existence of the mapping $\Delta_d : H'_d \times H'_d \to \mathbb{R}_+$. Consider a bipartite graph with nodes $L = \{\ell_e : e \in H'_d\}$ and $R = \{r_f : f \in H'_d\}$, and arcs $L \times R$. There is an additional source node s, and arcs from s to each L-node and arcs from each R-node to s. Every arc (i,j) in this network has a *lower bound* $\alpha(i,j)$ and an *upper bound* $\beta(i,j)$. The goal is to find a *circulation* z such that $\alpha(i,j) \le z(i,j) \le \beta(i,j)$ for all arcs (i,j). Recall that circulation is an assignment of nonnegative values to the arcs of the network so that the in-flow equals the out-flow at every node. The lower/upper bounds are set as follows.

1. For each arc $(i,j) \in L \times R$, we have $\alpha(i,j) = 0$ and $\beta(i,j) = \infty$.
2. For each arc (s, ℓ_e) where $e \in H'_d$, we have $\alpha(s,\ell_e) = \beta(s,\ell_e) = x_e$.
3. For each arc (r_f, s), where $f \in H'_d$, we have $\alpha(r_f,s) = \beta(r_f,s) = z_f$.

Recall that $\boldsymbol{x}$ and $\boldsymbol{z}$ are the LP solutions corresponding to $\hat{v}(S_R)$ and $\hat{v}(S_O)$.

Given *any* circulation z, we define $\Delta_d(e,f) = z(\ell_e, r_f)$ for all $e,f \in H'_d$. Then, it is easy to see that all three conditions in Lemma 4 are satisfied.

It just remains to prove the existence of some circulation. By Hoffman's circulation theorem (Hoffman 2003), there is a circulation if and only if

$$\alpha\left(\delta^-(T)\right) \le \beta\left(\delta^+(T)\right), \qquad \forall T \text{ subset of nodes.} \tag{C.1}$$

Previously, $\delta^-(T)$ denotes all arcs from a node outside T to a node inside T; similarly, $\delta^+(T)$ denotes all arcs from a node inside T to a node outside T. This condition can be verified using the following cases:

- Case 1: $T \cap L \neq \emptyset$ and $T \cap R \neq R$. In this case, there is some arc from $L \times R$ in $\delta^+(T)$, so the RHS in (19) is ∞, which clearly satisfies the condition.
- Case 2: $T \cap L = \emptyset$. If source $s \notin T$ then $\alpha(\delta^-(T)) = 0$; so (C.1) is clearly true. If source $s \in T$ then $\beta(\delta^+(T)) \ge \sum_{e \in H'_d} x_e = 1$ as all of L lies outside T, and clearly $\alpha(\delta^-(T)) \le 1$; so (C.1) holds.
- Case 3: $T \cap R = R$. If $s \in T$ then $\alpha(\delta^-(T)) = 0$; so (C.1) is clearly true. If source $s \notin T$ then $\beta(\delta^+(T)) \ge \sum_{f \in H'_d} z_f = 1$ as all of R lies inside T, and clearly $\alpha(\delta^-(T)) \le 1$; so (C.1) holds.

C.3. Supplementary Results for MMO Algorithm

Recall that $\hat{v}(S_R) = \sum_\xi \hat{v}(S_R, \xi)$, where $\hat{v}(S_R, \xi)$ is defined as the LP in (4). Therefore, we can write $\hat{v}(S_R)$ as the following LP:

$$\begin{aligned}
\hat{v}(S_R) = \underset{x}{\text{maximize}} \ & \frac{1}{N} \sum_\xi \sum_{e \in E(\xi)} v_e x_e^\xi && && \text{(C.2)}\\
\text{s.t.} \ & \sum_{e \in E(\xi): j \in e} x_e^\xi \le 1 && \forall j \in D(\xi) \quad \forall \xi,\\
& \sum_{e \in E(\xi): i \in e} x_e^\xi \le 1 && \forall i \in S_A \cup S_B \quad \forall \xi,\\
& x_e^\xi = 0 && \forall e \sim S_A \setminus S_R \quad \forall \xi,\\
& x_e^\xi \ge 0 && \forall e \in E(\xi) \quad \forall \xi.
\end{aligned}$$

For any vehicle i and scenario ξ, set $F_{i,\xi} \subseteq E(\xi)$ denotes all the hyperedges incident to i in scenario ξ. All variables x_e^ξ with $e \sim S_A \setminus S_R$ are set to zero. Therefore, it suffices to consider the LP with variables x_e^ξ for $e \in F_{i,\xi}$ and $i \in S_B \cup S_R$.

We now consider the dual of the above LP (which has the same optimal value by strong duality). Let $G = S_A \cup S_B \cup (\cup_\xi D(\xi))$ denote a combined groundset consisting of *all* vehicles and demands from all scenarios. The dual variables are $u_{g,\xi}$ for all $g \in G$ and scenarios ξ. The dual LP is

$$\begin{aligned}
\hat{v}(S_R) \ = \ \underset{u}{\text{minimize}} \ & \sum_\xi \sum_{g \in G} u_{g,\xi}\\
\text{s.t.} \ & \sum_{g \in e} u_{g,\xi} \ge \frac{v_e}{N},\\
& \forall e \in F_{i,\xi}, \quad \forall \xi, \quad \forall i \in S_R \cup S_B\\
& \boldsymbol{u} \ge 0.
\end{aligned}$$

References

Alonso-Mora J, Samaranayake S, Wallar A, Frazzoli E, Rus D (2017) On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proc. National Acad. Sci. USA* 114(3):462–467.

Arkin EM, Hassin R (1998) On local search for weighted k-set packing. *Math. Oper. Res.* 23(3):640–648.

Ashlagi I, Burq M, Dutta C, Jaillet P, Saberi A, Sholley C (2018) Maximum weight online matching with deadlines. Preprint, submitted August 9, https://arxiv.org/abs/1808.03526.

Benjaafar S, Wu S, Liu H, Gunnarsson EB (2021) Dimensioning on-demand vehicle sharing systems. *Management Sci.* 68(2): 1218–1232.

Bertsimas D, Tsitsiklis JN (1997) *Introduction to Linear Optimization*, volume 6 (Athena Scientific, Belmont, MA).

Bollobás B, Erdös P (1976) Cliques in random graphs. *Math. Proc. Cambridge Philosophical Soc.* 80:419–427.

Braverman A, Dai JG, Liu X, Ying L (2019) Empty-car routing in ridesharing systems. *Oper. Res.* 67(5):1437–1452.

Buchbinder N, Chen S, Gupta A, Nagarajan V, Naor J (2014) Online packing and covering framework with convex objectives. Preprint, submitted December 29, https://arxiv.org/abs/1412.8347.

Castro F, Frazier P, Ma H, Nazerzadeh H, Yan C (2020) Matching queues, flexibility and incentives. Preprint, submitted June 16, https://arxiv.org/abs/2006.08863.

Chan YH, Lau LC (2012) On linear and semidefinite programming relaxations for hypergraph matching. *Math. Programming* 135(1):123–148.

Chekuri C, Khanna S (2005) A polynomial time approximation scheme for the multiple knapsack problem. *SIAM J. Comput.* 35(3):713–728.

Chen D, Ahn S, Chitturi M, Noyce DA (2017a) Toward vehicle automation: Roadway capacity formulation for traffic mixed with regular and automated vehicles. *Transportation Res. Part B: Methodological* 100:196–221.

Chen Z, He F, Yin Y, Du Y (2017b) Optimal design of autonomous vehicle zones in transportation networks. *Transportation Res. Part B: Methodological* 99:44–61.

Data NO (2022) Nyc community district data. Accessed November 2, 2021, https://data.cityofnewyork.us/City-Government/Community-Districts/yfnk-k7r4.

Dong J, Ibrahim R (2020) Managing supply in the on-demand economy: Flexible workers, full-time employees, or both? *Oper. Res.* 68(4):1238–1264.

Dong T, Sun X, Luo Q, Wang J, Yin Y (2023) The dual effects of team contest design on on-demand service work schedules. *Service Sci.*, ePub ahead of print March 8, https://doi.org/10.1287/serv.2023.0320.

Dong T, Xu Z, Luo Q, Yin Y, Wang J, Ye J (2021) Optimal contract design for ride-sourcing services under dual sourcing. *Transportation Res. Part B: Methodological* 146:289–313.

Feige U, Jain K, Mahdian M, Mirrokni V (2007) Robust combinatorial optimization with exponential scenarios. Fischetti M, Williamson DP, eds. *Proc. Internat. Conf. on Integer Programming and Combinatorial Optim.* Lecture Notes in Computer Science, vol. 4513 (Springer, Berlin), 439–453.

Fleischer L, Goemans MX, Mirrokni VS, Sviridenko M (2006) Tight approximation algorithms for maximum general assignment problems. *Proc. 17th Annual ACM-SIAM Sympos. on Discrete Algorithm*, 611–620.

Füredi Z, Kahn J, Seymour PD (1993) On the fractional matching polytope of a hypergraph. *Combinatorica* 13(2):167–180.

Geng X, Li Y, Wang L, Zhang L, Yang Q, Ye J, Liu Y (2019) Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting. *Proc. Conf. AAAI Artificial Intelligence* 33:3656–3663.

Guda H, Subramanian U (2019) Your uber is arriving: Managing on-demand workers through surge pricing, forecast communication, and worker incentives. *Management Sci.* 65(5):1995–2014.

Guo X, Caros NS, Zhao J (2021) Robust matching-integrated vehicle rebalancing in ride-hailing system with uncertain demand. *Transportation Res. Part B: Methodological* 150:161–189.

Gupta A, Nagarajan V (2014) Approximating sparse covering integer programs online. *Math. Oper. Res.* 39(4):998–1011.

Gupta A, Nagarajan V, Ravi R (2015) Robust and maxmin optimization under matroid and knapsack uncertainty sets. *ACM Trans. Algorithms* 12(1):1–21.

Hasan MH, Van Hentenryck P (2021) The benefits of autonomous vehicles for community-based trip sharing. *Transportation Res., Part C Emerging Tech.* 124:102929.

Hasan MH, Van Hentenryck P, Legrain A (2020) The commute trip-sharing problem. *Transportation Sci.* 54(6):1640–1675.

Hazan E, Safra S, Schwartz O (2006) On the complexity of approximating k-set packing. *Comput. Complexity* 15(1):20–39.

Herminghaus S (2019) Mean field theory of demand responsive ride pooling systems. *Transportation Res. Part A Policy Practices* 119:15–28.

Hoffman AJ (2003) Inequalities to extremal combinatorial analysis. *Selected Papers Alan Hoffman Commentary* 10:244.

Karamanis R, Anastasiadis E, Stettler M, Angeloudis P (2021) Vehicle redistribution in ride-sourcing markets using convex minimum cost flows. *IEEE Trans. Intelligent Transportation Systems.* 23(8):10287–10298.

Ke J, Zheng Z, Yang H, Ye J (2021) Data-driven analysis on matching probability, routing distance and detour distance in ride-pooling services. *Transporation Res., Part C Emerging Tech.* 124:102922.

Lavieri PS, Bhat CR (2019) Modeling individuals' willingness to share trips with strangers in an autonomous vehicle future. *Transportation Res. Part A Policy Practice* 124:242–261.

Li J, Chen D, Zhang M (2022) Equilibrium modeling of mixed autonomy traffic flow based on game theory. *Transportation Res. Part B: Methodological* 166:110–127.

Li S, Luo Q, Hampshire RC (2021) Optimizing large on-demand transportation systems through stochastic conic programming. *Eur. J. Oper. Res.* 295(2):427–442.

Liu Y, Samaranayake S (2020) Proactive rebalancing and speed-up techniques for on-demand high capacity ridesourcing services. *IEEE Trans. Intelligent Transportation Systems.* 23(2):819–826.

Lokhandwala M, Cai H (2018) Dynamic ride sharing using traditional taxis and shared autonomous taxis: A case study of NYC. *Transportation Res., Part C Emerging Tech.* 97:45–60.

Lowalekar M, Varakantham P, Jaillet P (2020) Competitive ratios for online multi-capacity ridesharing. *Proc. 19th Internat. Conf. on Autonomous Agents and MultiAgent Systems*, 771–779.

Ma J, Li X, Zhou F, Hao W (2017) Designing optimal autonomous vehicle sharing and reservation systems: A linear programming approach. *Transporation Res., Part C Emerging Tech.* 84:124–141.

Markov I, Guglielmetti R, Laumanns M, Fernández-Antolín A, de Souza R (2021) Simulation-based design and analysis of on-demand mobility services. *Transportation Res. Part A Policy Practice* 149:170–205.

Matula DW (1976) *The Largest Clique Size in a Random Graph* (Department of Computer Science, Southern Methodist University, Dallas, TX).

Mori JCM, Samaranayake S (2021) On the request-trip-vehicle assignment problem. *Proc. SIAM Conf. on Appl. and Comput. Discrete Algorithms*, 228–239.

Nemhauser GL, Wolsey LA, Fisher ML (1978) An analysis of approximations for maximizing submodular set functions—i. *Math. Programming* 14(1):265–294.

Öncan T (2007) A survey of the generalized assignment problem and its applications. *INFOR Inform. Systems Oper. Res.* 45(3): 123–141.

Pagnoncelli BK, Ahmed S, Shapiro A (2009) Sample average approximation method for chance constrained programming: Theory and applications. *J. Optim. Theory Appl.* 142(2): 399–416.

Qin G, Luo Q, Yin Y, Sun J, Ye J (2021) Optimizing matching time intervals for ride-hailing services using reinforcement learning. *Transportation Res. Part C Emerging Tech.* 129:103239.

Qin ZT, Zhu H, Ye J (2021) Reinforcement learning for ridesharing: A survey. *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)* (IEEE, Piscataway, NJ), 2447–2454.

Santi P, Resta G, Szell M, Sobolevsky S, Strogatz SH, Ratti C (2014) Quantifying the benefits of vehicle pooling with shareability networks. *Proc. National Acad. Sci. USA* 111(37):13290–13294.

Shah S, Lowalekar M, Varakantham P (2020) Neural approximate dynamic programming for on-demand ride-pooling. *Proc. Conf. AAAI Artificial Intelligence* 34:507–515.

Shladover SE (2018) Connected and automated vehicle systems: Introduction and overview. *J. Intelligent Transportation Systems* 22(3):190–200.

Shmoys DB, Tardos É (1993) An approximation algorithm for the generalized assignment problem. *Math. Programming* 62(1–3):461–474.

Simonetto A, Monteil J, Gambella C (2019) Real-time city-scale ridesharing via linear assignment problems. *Transportation Res., Part C Emerging Tech.* 101:208–232.

Sundt A, Luo Q, Vincent J, Shahabi M, Yin Y (2021) Heuristics for customer-focused ride-pooling assignment. Preprint, submitted July 23, https://arxiv.org/abs/2107.11318.

Tang X, Qin Z, Zhang F, Wang Z, Xu Z, Ma Y, Zhu H, et al. (2019) A deep value-network based approach for multi-driver order dispatching. *Proc. 25th ACM SIGKDD Internat. Conf. on Knowledge Discovery and Data Mining*, 1780–1790.

TLC (2021) Nyc taxi and limousine commission trip record data. Accessed May 2, 2021, https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page.

Wang H, Yang H (2019) Ridesourcing systems: A framework and review. *Transportation Res. Part B: Methodological* 129:122–155.

Wei Q, Pedarsani R, Coogan S (2020) Mixed autonomy in ride-sharing networks. *IEEE Transportation Control Network Systems* 7(4):1940–1950.

Xie J, Liu Y, Chen N (2023) Two-sided deep reinforcement learning for dynamic mobility-on-demand management with mixed autonomy. *Transportation Sci.*, ePub ahead of print January 17, https://doi.org/10.1287/trsc.2022.1188.

Yang H, Qin X, Ke J, Ye J (2020) Optimizing matching time interval and matching radius in on-demand ride-sourcing markets. *Transportation Res. Part B: Methodological* 131:84–105.

Yu X, Shen S (2019) An integrated decomposition and approximate dynamic programming approach for on-demand ride pooling. *IEEE Trans. Intelligent Transportation Systems* 21(9): 3811–3820.