MATE: A Memory-Augmented Time-Expansion Approach for Optimal Trip-Vehicle Matching and Routing in Ride-Sharing

Ye Tian
Dept. of Computer Science
Iowa State University
Ames, Iowa, USA

Jia Liu*
Dept. of Computer Science
Iowa State University
Ames, Iowa, USA

Cathy Xia†
Dept. of Integrated Systems Engineering
The Ohio State University
Columbus, Ohio, USA

ABSTRACT

Spurred by increasing fuel shortage and environmental concerns, ride-sharing systems have attracted a great amount of attention in recent years. Lying at the heart of most ride-sharing systems is the problem of joint trip-vehicle matching and routing optimization, which is highly challenging and results in this area remain rather limited. This motivates us to fill this gap in this paper. Our contributions in this work are three-fold: i) We propose a new analytical framework that jointly considers trip-vehicle matching and optimal routing; ii) We propose a linearization reformulation that transforms the problem into a mixed-integer linear program, for which moderate-sized instances can be solved by global optimization methods; and iii) We develop a memory-augmented time-expansion (MATE) approach for solving large-sized problem instances, which leverages the special problem structure to facilitate approximate (or even exact) algorithm designs. Collectively, our results advance the state-of-the-art of intelligent ride-sharing and contribute to the field of sharing economy.

CCS CONCEPTS

• **Applied computing → Transportation**; • **Theory of computation → Mathematical optimization**; **Approximation algorithms analysis**; • **Hardware → Fuel-based energy**.

KEYWORDS

Ride-Sharing, optimization, approximation algorithm

ACM Reference Format:
Ye Tian, Jia Liu, and Cathy Xia. 2020. MATE: A Memory-Augmented Time-Expansion Approach for Optimal Trip-Vehicle Matching and Routing in Ride-Sharing. In *The Eleventh ACM International Conference on Future Energy Systems (e-Energy'20), June 22–26, 2020, Virtual Event, Australia.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3396851.3397726

*J. Liu's work is supported in part by NSF grants CAREER CNS-1943226, ECCS-1818791, CCF-1758736, CNS-1758757, and a Google Faculty Research Award.
†C. Xia's work is supported in part by NSF grant SES-1409214 and a Ford OSU Alliance Research Award.

e-Energy'20, June 22–26, 2020, Virtual Event, Australia

ACM ISBN 978-1-4503-8009-6/20/06…$15.00
https://doi.org/10.1145/3396851.3397726

1 INTRODUCTION

In recent years, spurred by increasing fuel shortage and environmental concerns on heavy fossil fuels uses, ride-sharing systems have attracted great attention [38]. Ride-sharing enables multiple passengers with similar itineraries and time schedules to share a vehicle to alleviate multiple societal problems, e.g., traffic congestion, environmental pollution, etc. [1] For instance, studies in [2, 27] show that, with ride-sharing, 3,000 vehicles of capacity four are sufficient to serve 98% of the Manhattan ride requests (currently served by over 13,000 taxis) with marginal increment in the trip delay. The total trip distance (a surrogate metric for commute time and gas consumption) could also be reduced by more than 30%. This leads to a *win-win* solution between customers and service providers: Passengers enjoy reduced payments at the expense of tolerating extra delay, while drivers earn higher income by completing multiple transactions in a single outing. Moreover, ride-sharing achieves benefits similar to public transportation, while offering *flexible routes* and *infinitesimal granularity* in service coverage.

However, designing efficient ride-sharing system is non-trivial because poor ride-sharing decisions could lead to either low vehicle occupancy or excessive delays. To boost vehicle occupancy and reduce delay in ride-sharing, two of the most fundamental design elements are *trip-vehicle matching* and *vehicle routing*, i.e., (i) How to appropriately group riders and assign them to vehicles; and (ii) Finding a fuel-efficient path for the driver to deliver the assigned riders. Unfortunately, as will be shown later, the joint trip-vehicle matching and vehicle routing optimization is a mixed-integer nonlinear program (MINLP), which is not only NP-Hard [39] but also not amenable for existing approximation algorithm techniques. To date, existing algorithmic results on optimal trip-vehicle matching and vehicle routing with optimality guarantee remains rather limited (see Section 2 for in-depth discussions). In light of the increasing relevance of ride-sharing, there is a compelling need to fill this gap and rigorously investigate the design of optimal trip-vehicle matching and vehicle routing for ride-sharing systems.

In this paper, we address the above challenges by proposing a series of new global and approximation optimization methodologies. Our main contributions are summarized as follows:

- We propose a new analytical framework that captures the most essential features of trip-vehicle matching and vehicle routing to enable rigorous optimization algorithm for ride-sharing. Our proposed analytical framework includes (i) a product-form model

[1] For example, the annual congestion cost in Manhattan is over $20 billion [34] (including 500 million gallons of fuel consumed by 24 million hours of traffic idling).

that simultaneously characterizes riders' time window feasibility and trip-vehicle matching, (ii) a flow-balance model that allows dynamic vehicle routing optimization, and (iii) versatile utility model that incorporates both drivers' earning and fuel consumption. We then show that, under this analytical framework, trip-vehicle matching and vehicle routing can be unified and jointly formulated as an MINLP problem.

- To overcome the fundamental hardness in solving the formulated MINLP problem, we take a two-pronged approach. First, through a clever logically-equivalent reformulation, we transform the original MINLP problem into a mixed-integer linear program (MILP). Thanks to this reformulation, moderate-sized ride-sharing problems become well solvable by existing optimization solvers (e.g., Gurobi[13], CPLEX [1], Mosek [3] etc.). This global optimization approach also serves as a baseline for performance evaluation in our subsequent algorithmic design.
- Note, however, that the complexity of solving MILP remains NP-Hard and does not scale well for large-sized problem instances. To address this challenge, we propose a customized *memory-augmented time-expansion* (MATE) approach, which exploits the special graphical structure of the joint trip-vehicle matching and vehicle routing problem in both space and time domains. Under the proposed MATE approach, single-vehicle ride-sharing problems can be solved *exactly* thanks to an inherent total unimodularity (TU) property [5]. For multi-vehicle ride-sharing problems, our MATE approach enables the use of state-of-the-art approximation techniques for multi-commodity network flows to solve ride-sharing problems with strong performance guarantee.

Collectively, our results advance intelligent ride-sharing and contribute to the future prospects of sharing economy. The rest of this paper is organized as follows. In Sec. 2, we review related work and put this paper into comparative perspectives. In Sec. 3, we present the network model and the MINLP problem formulation. In Sec. 4, we introduce our global optimization approach, the MILP reformulation. Sec. 5 is focused on presenting the graphical construction in our MATE approach, which is followed by algorithm designs for single- and multi-vehicle settings in Sec. 6. Sec. 7 illustrates numerical results and Sec. 8 concludes this paper.

2 RELATED WORK

Historically, the idea of ride-sharing traces its root to the Operations Research (OR) literature dating back to 1940s during the World War II, but rapidly takes off in recent years thanks to the proliferation of smart mobile devices. In the Operations Research (OR) literature, the canonical ride-sharing problem formulation is referred to as the PDPTW problem (pickup and delivery problem with time windows) [41], which considers time, routing and capacity constraints [4, 7–9, 11, 14, 16–18, 22, 24–26, 32, 33, 35, 36, 41]. Broadly speaking, solution methods for PDPTW problems can be categorized as heuristic [4, 7–9, 16–18, 24–26, 32, 33, 35, 36, 41] and exact [8, 9, 11, 14, 22, 32, 33] approaches. Common methods in the heuristic category include, e.g., genetic algorithms [15], adaptive local neighborhood search [28], Lagrangian decomposition [28], etc., which do *not* provide any optimality guarantee in general.

In comparison, the basic idea of exact approaches is to leverage global optimization techniques (e.g., branch-and-cut, column generation, etc.), which guarantee solving PDPTW problems to optimality. However, due to the NP-Hardness of PDPTW problems, these exact approaches often do not scale well and can only handle moderate-sized problems. In the exact approach category, the most related work to ours is [37], where Ropke *et al.* proposed a similar linearization techniques to transform the PDPTW problems into an MILP formulation, which is then solved by the branch-and-cut method. However, the model in [37] is only limited to the single-vehicle setting, while our work also considers the multi-vehicle setting. As a result, the problem dimensionality in our work is significantly higher and necessitates efficient approximation solutions. Surprisingly, for the single-vehicle setting, our MATE approach *even* provides a polynomial-time *exact* solution by exploiting the special structure of the capacity-constrained PDPTW problem.

By contrast, in the computing literature where computational complexity and efficiency are of utmost concern, various efficient approximation algorithms with performance guarantee have been proposed for ride-sharing problems. However, most of these existing work considered alternative formulations that are quite different from PDPTW. For example, in [20], Lin *et al.* proposed a pseudo-polynomial-time dynamic programming approach to solve a two-stage route planning problem to maximize the expected total revenue in a single-driver setting. In their follow-up work [21], another route planning problem was formulated to maximize the expected number of riders picked up by multiple vehicles, which was solved by a $(1 - 1/\mathrm{e})$ approximation algorithm based on dual sub-gradient decent. The most related work to ours in the area of approximation algorithm design is [6], where Bei and Zhang developed a 2.5-approximation algorithm based on minimum weight matching to jointly determine ride matching and routing for drivers' cost minimization. However, riders' time window feasibility, a key feature in PDPTW, were not considered in [6]. In contrast, our work directly tackles the canonical PDPTW model, where all three key elements, i.e., time window, capacity and routing constraints are jointly considered. To our knowledge, the proposed MATE approach is the first polynomial-time method that guarantees an *exact* solution in the capacity-constrained single-vehicle setting and an approximation ratio for the multi-vehicle setting.

3 NETWORK MODEL AND RIDE-SHARING PROBLEM FORMULATION

In this section, we present the network model and the problem formulation. Specifically, we first illustrate in Section 3.1 the process of constructing a virtual graph for a given physical road network, which simplifies the subsequent problem formulation. In Section 3.2, we present each key component of our ride-sharing problem, including time window feasibility, capacity and routing constraints.

3.1 A Virtual Graph Modeling Approach

In this paper, we introduce a virtual graph modeling approach for physical networks to facilitate efficient ride-sharing problem formulation. Consider a physical road network represented by an undirected graph $\mathcal{G}_0 \ , \ (\mathcal{N}_0; \mathcal{L}_0)$, where $\mathcal{N}_0$ represents the sets of geographic locations, $\mathcal{L}_0$ represents roads between nodes in $\mathcal{N}_0$. We assume $\mathcal{G}_0$ is connected. Let $\mathcal{R}$ and $\mathcal{V}$ denote the sets of all rider groups and drivers with size $|\mathcal{R}|$ and $|\mathcal{V}|$, respectively.

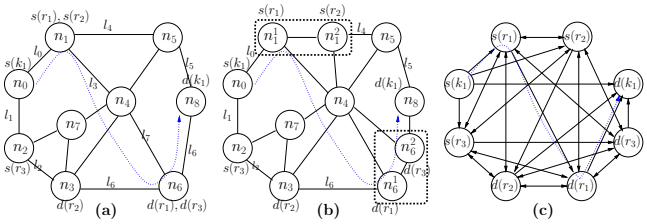

Figure 1: (a) Original graph for the physical road network, (b) Intermediate graph in Step 1, (c) Virtual graph in Step 2.

We assume that every rider group $r \in \mathcal{R}$ only has one pickup location and one drop-off location. Let $s(\cdot)$ and $d(\cdot)$ be the source and destination nodes of a rider group or a vehicle driver, respectively. As an example, Fig. 1a shows a physical road network $\mathcal{G}_0$, where the directed dashed line represent the route for rider groups r_1. For each edge $l_{i;j}$, $i; j \in \mathcal{N}_0$, we associate two parameters $t_{i;j}$ and $\delta_{i;j}$ to represent the travel time and distance between nodes i and j, respectively. For modeling convenience, $t_{i;j}$ includes service time overhead (e.g., time for getting on/off a vehicle) when the vehicle reaches node j. Our virtual graph construction is a two-step process (which, to our knowledge, is new in the literature):

Step 1): Our first step is to resolve a modeling ambiguity over a given physical graph $\mathcal{G}_0$. For example, as shown in Fig. 1a, if two rider groups r_1 and r_2 all start from node n_1 and two rider groups r_1 and r_3 share the same destination node n_6, then it is unclear whether a planning path starting from n_1 corresponds to rider group r_1 or r_2, and whether a planning path ending at node n_6 corresponds to rider group r_1 or r_3. This can be resolved as follows: If there are h rider groups and vehicles sharing the same geographic location, then we construct h virtual nodes with zero distance between each pair of them. As an example, in Fig. 1b, we create two virtual nodes n_1^1 ($s(r_1)$) and n_1^2 ($s(r_2)$) to replace the physical node n_1, and two virtual nodes n_6^1 ($d(r_1)$) and n_6^2 ($d(r_3)$) to replace the physical node n_6. The distances between all co-located virtual nodes are zero.

Step 2): We use $\mathcal{N}$ to denote the set of all virtual nodes that correspond to sources and destinations of all rider groups and all vehicles. That is, intermediate physical nodes that are neither source nor destination nodes of any rider group or vehicle are removed. We then construct a set of directed edges $\mathcal{L}$ as follows: For any rider group $r_i \in \mathcal{R}$, we add a directed edge from virtual node $s(r_i)$ to virtual node $d(r_i)$. The directed edge $s(r_i) \rightarrow d(r_i)$ is associated with parameters $t_{s(r_i);d(r_i)}$ and $d_{s(r_i);d(r_i)}$, which correspond to the travel time and distance of a *shortest path* between $s(r_i)$ and $d(r_i)$ in the physical network. The direction of this edge means that it cannot travel from $d(r_i)$ to $s(r_i)$. Next, for all $i; j \in \mathcal{R}$ with $i \neq j$, we add a pair of directed edges $s(r_i) \rightarrow d(r_j)$ and $d(r_j) \rightarrow s(r_i)$. Both of these directed edges are associated with parameters $t_{s(r_i);d(r_j)}$ and $d_{s(r_i);d(r_j)}$, which represent the travel time and distance of a shortest path between $s(r_i)$ and $d(r_j)$ in the physical network [2]. For each vehicle driver $v_k \in \mathcal{V}$, we add directed edges $s(v_k) \rightarrow d(v_k)$, $s(v_k) \rightarrow s(r_i)$ and $d(r_i) \rightarrow d(v_k)$, $\forall r_i \in \mathcal{R}$. For the physical road network in Fig. 1a, the final virtual graph is illustrated in Fig. 1c.

[2]For simplicity, in this paper, we assume that the road networks are symmetric so that $t_{i;j} = t_{j;i}$, $d_{i;j} = d_{j;i}$. Our algorithms and results can also be straightforwardly extended to asymmetric cases at the expense of slightly more complex notation.

3.2 Ride-sharing Problem Formulation

With the above virtual graph model, we are in a position to formally define our ride-sharing problem.

1) Vehicle Routing Constraints: In this paper, a path taken by a vehicle k consists of a sequence of edges in the virtual graph. For example, the path of vehicle k passing through r_1's source and destination and reaching its own destination can be represented as:

$$s(k) \xrightarrow{\text{link } (s(k);s(r_1))} s(r_1) \xrightarrow{\text{link } (s(r_1);d(r_1))} d(r_1) \xrightarrow{\text{link } (d(r_1);d(k))} d(k):$$

To model the path selection, we let a binary variable $x_{i;j}^k = 1$ if vehicle k's path contains edge $l_{i;j}$ and $x_{i;j}^k = 0$ otherwise. To avoid triviality, we let $x_{i;i}^k \equiv 0; \forall i \in \mathcal{N}; \forall k \in \mathcal{V}$. Since vehicle k can only head for one node at its source and arrive from one node at its destination, we have:

$$\sum_{j \in \mathcal{N} \setminus \{s(k)\}} x_{s(k);j}^k = 1; \qquad \sum_{i \in \mathcal{N} \setminus \{d(k)\}} x_{i;d(k)}^k = 1; \forall k \in \mathcal{V}: \tag{1}$$

In this paper, we consider Uber-like systems where riders do not change vehicle and transfer. As a result, every rider group r can only be matched to at most one vehicle if he/she is picked up. This can be modeled as:

$$\sum_{j \in \mathcal{N} \setminus \{i\}} x_{i;j}^k \leq 1; \forall i \in \mathcal{N}: \tag{2}$$

Also, both the pickup and drop-off of a rider group r should be performed by the same vehicle k, which can be modeled as:

$$\sum_{j \in \mathcal{N}; j \neq s(r)} x_{s(r);j}^k = \sum_{j \in \mathcal{N}; j \neq d(r)} x_{j;d(r)}^k; \forall r \in \mathcal{R}; \forall k \in \mathcal{V}: \tag{3}$$

We assume that once vehicle $v_k \in \mathcal{V}$ departs from its source $s(v_k)$, it will continue to provide service until arriving at its destination $d(v_k)$. In other words, any vehicle arriving at an intermediate node must also leave this node. Therefore, the following flow-balance-type constraint holds for each vehicle:

$$\sum_{i \in \mathcal{N} \setminus \{u\}} x_{i;u}^k = \sum_{j \in \mathcal{N} \setminus \{u\}} x_{u;j}^k; \forall u \in \mathcal{N} \setminus \{s(k); d(k)\}; \forall k \in \mathcal{V}: \tag{4}$$

2) Time Window Feasibility Constraints: We use μ_n and λ_n, $\forall n \in \mathcal{N}$, to define the earliest arrival time and latest departure time of a virtual node n. Then, we can compute a rider group r_i's departure and arrival time windows as: $\mu_{s(r_i)}; \lambda_{d(r_i)} - t_{s(r_i);d(r_i)}$; $\mu_{s(r_i)} + t_{s(r_i);d(r_i)}; \lambda_{d(r_i)}$. By the same token, a vehicle v_k's departure and arrival time windows can be computed as $\mu_{s(v_k)}; \lambda_{d(v_k)} - t_{s(v_k);d(v_k)}$; $\mu_{s(v_k)} + t_{s(v_k);d(v_k)}; \lambda_{d(v_k)}$. We let τ_i^k represent vehicle k's departure time at virtual node i. For any two consecutive nodes i and j in the path taken by vehicle k, the departure time at node j can be computed as $\tau_j^k = \tau_i^k + t_{i;j}$. We can further model these two facts jointly as a single constraint:

$$x_{i;j}^k \left(\tau_i^k + t_{i;j} - \tau_j^k \right) = 0; \forall i; j \in \mathcal{N}; i \neq j: \tag{5}$$

If vehicle k picks up rider group r, then the following holds:

$$\mu_{s(r)} \leq \tau_{s(r)}^k \leq \tau_{d(r)}^k \leq \lambda_{d(r)}; \forall r \in \mathcal{R}; \tag{6}$$

i.e., rider group r should arrive at $s(r)$ [$d(r)$] earlier [later] than vehicle k does. The same constraint is also true for every vehicle k:

$$\mu_{s(k)} \leq \tau_{s(k)}^k \leq \tau_{d(k)}^k \leq \lambda_{d(k)}; \forall k \in \mathcal{V}; \tag{7}$$

i.e., vehicle k should arrive at $s(k)$ [$d(k)$] earlier [later] than its departure [arrival] time.

3) Vehicle Capacity and Load Change Constraints: We let y_i^k denote the load of vehicle k, i.e, number of passengers on-board, immediately after serving then leaving virtual node i. We let ρ_i be the load change at node i, i.e., the number of riders that will get on/off at this virtual node. Then, the coupling between routing decisions and load change dynamics can be written as:

$$x_{i,j}^k \left(y_i^k + \rho_j - y_j^k \right) = 0, \forall i, j \in \mathcal{N}, i \neq j, \tag{8}$$

i.e., if a vehicle k travels from node i to node j, then the difference between y_j^k and y_i^k is exactly ρ_j. Also, note that the number of riders in a rider group r getting on a vehicle at node $s(r)$ must all get off at node $d(r)$. This implies $\rho_{s(r)} + \rho_{d(r)} = 0, \forall r \in \mathcal{R}$. Since the capacity of each vehicle cannot be exceeded, we have:

$$\max\{\rho_i, 0\} \leq y_i^k \leq P^k, \forall i \in \mathcal{R}_a, \tag{9}$$

where P^k is vehicle k's capacity limit, $\mathcal{R}_a = \mathcal{R}_s \cup \mathcal{R}_d$ is the set of all rider group sources and destinations. According to the way we construct the virtual graph, no rider groups get on and get off at vehicles' source and destination nodes, which implies:

$$y_{s(k)}^k = y_{d(k)}^k = 0, \forall k \in \mathcal{V}. \tag{10}$$

4) Objective Function and Problem Formulation: In Uber-like ride-sharing systems, maximizing the profits earned by vehicle drivers is important because it attracts more vehicles to participate in ride-sharing and ensures the sustainability of the ride-sharing system. In this paper, we follow a price model adopted by most ride-sharing systems in practice. We assume that the reward of accepting the transaction of rider group r is proportional to the direct distance between $s(r)$ and $d(r)$, *regardless* of how many detours the vehicle takes to serve rider group r. This is reasonable because no rider group is willing to pay extra cost for detours in ride-sharing. Thus, the reward of serving rider group r can be computed as $\beta \cdot d_{s(r),d(r)}$ where β is the unit reward rate of serving a rider group. Let α be the vehicle cost rate per mile due to the combined effect of fuel consumption, maintenance, toll fees, etc. Let $u_{i,j}$ denote the net profit of a vehicle travelling from node i to node j. Based on the above modeling, $u_{i,j}$ can be computed as:

$$u_{i,j} = \begin{cases} -\alpha d_{i,j} + \beta d_{s(r),d(r)}, & j = s(r), \text{ for some } r \in \mathcal{R}; \\ -\alpha d_{i,j}, & \text{otherwise.} \end{cases} \tag{11}$$

Intuitively, Eq. (11) means that a vehicle earns profit traveling from node i to node j only if the driver picks up some rider group r at node j; otherwise, traveling from i to j only incurs cost. Note that $u_{i,j}$ can be pre-computed. With the above modeling, our profit maximization ride-sharing problem (PMRS) can be formulated as:

$$\textbf{PMRS: } \max_{x} \sum_{k \in \mathcal{V}} \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} x_{i,j}^k u_{i,j}$$

subject to Constraints (1)–(10):

We note that Problem PMRS is a mixed-integer nonlinear programming (MINLP) problem due to the product terms in (5) and (8). As a result, not only is Problem PMRS non-convex, it also has a complex structure that cannot be directly handled by existing optimization solvers. To address this challenge, we propose a two-pronged approach. In Sec. 4, we develop a global optimization approach based on a *logically-equivalent linear reformulation*. Then, for large-sized problem instances, we develop an approximation solution based on a customized memory-augmented time-expansion approach (MATE) in Sec. 5. For convenience, we summarize the key notation of this paper in Table 1.

Table 1: Notation.

Symbol	Meaning
$\mathcal{R}$	The set of all rider groups
r_i	Rider group r_i
$\mathcal{V}$	The set of all vehicles
v_k	Vehicle v_k
$\mathcal{R}_s$	The set of rider groups' pickup nodes
$\mathcal{R}_d$	The set of rider groups' delivery nodes
$\mathcal{R}_a$	The set of all pick-up and drop-off nodes
$\mathcal{N}$	The set of all virtual nodes
$t_{i,j}$ / $\delta_{i,j}$	The shortest(direct) travel time/distance between nodes i and j, $(i,j) \in \mathcal{L}$
ρ_i	The load change at node i
y_i^k	The load after service node i by vehicle k
τ_i^k	The departure time at node i by vehicle k
P^k	The maximum capacity of a vehicle k
$x_{i,j}^k$	Vehicle k's routing decision variable. $x_{i,j}^k = 1$ if vehicle k chooses edge (i,j), otherwise $x_{i,j}^k = 0$
μ_n / λ_n	The earliest/latest departure time (if n is $s(\cdot)$) or arrival time(if n is $d(\cdot)$)

4 A GLOBAL OPTIMIZATION APPROACH

As mentioned earlier, Problem PMRS is an MINLP and the main difficulty in solving it stems from the mixed-integer product-form constraints in (5) and (8). Our basic idea to address this challenge is to reformulate and linearize (5) and (8). This transforms PMRS into a mixed-integer linear programming (MILP) that is directly solvable by existing optimization solvers (e.g., CPLEX, Gurobi, etc.).

Toward this end, we start with (5), which can be equivalently rewritten as the following two inequalities:

$$x_{i,j}^k \left(\tau_i^k + t_{i,j} - \tau_j^k \right) \leq 0, \tag{12}$$

$$x_{i,j}^k \left(\tau_i^k + t_{i,j} - \tau_j^k \right) \geq 0. \tag{13}$$

Noting from constraint (6) that the values of τ_i^k-variables are finite, we let LB_k and UB_k be the lower and upper bounds of $(\tau_i^k + t_{i,j} - \tau_j^k)$. Then, Eqs. (12) and (13) are, respectively, logically equivalent to:

$$\tau_i^k + t_{i,j} - \tau_j^k - (1 - x_{i,j}^k) UB_k \leq 0, \tag{14}$$

$$\tau_i^k + t_{i,j} - \tau_j^k - (1 - x_{i,j}^k) LB_k \geq 0. \tag{15}$$

To see this equivalence, note that if $x_{i,j}^k = 1$, Eqs. (14) and (15) are identical to (12) and (13). Otherwise, if $x_{i,j}^k = 0$, Eqs. (14) and (15) imply $LB_k \leq \tau_i^k + t_{i,j} - \tau_j^k \leq UB_k$, which trivially holds following from the definition of LB_k and UB_k. To obtain LB_k and UB_k, we note that $\tau_i^k \in [\mu_i, \lambda_i]$, $\tau_j^k \in [\mu_j, \lambda_j]$. It then follows that UB_k and LB_k can be chosen as $UB_k = \lambda_i + t_{i,j} - \mu_j$ and $LB_k = \mu_i + t_{i,j} - \lambda_j$.

As a result, Eq. (5) can be reformulated as the following two mixed-integer linear constraints:

$$\tau_i^k + t_{i,j} - \tau_j^k \;-\; 1 - x_{i,j}^k \quad \lambda_i + t_{i,j} - \mu_j \;\le 0; \tag{16}$$

$$\tau_i^k + t_{i,j} - \tau_j^k \;-\; 1 - x_{i,j}^k \quad \mu_j + t_{i,j} - \lambda_i \;\ge 0; \tag{17}$$

for all $i; j \in \mathcal{N}; i \neq j$. We can apply the same reformulation and linearization technique to Eq. (8) by first equivalently rewriting it as the following two inequalities:

$$x_{i,j}^k \;\; y_i^k + \rho_j - y_j^k \;\le 0; \tag{18}$$

$$x_{i,j}^k \;\; y_i^k + \rho_j - y_j^k \;\ge 0: \tag{19}$$

The vehicle capacity limit constraint in (9) naturally provides lower and upper bounds for the term $(y_i^k + \rho_j - y_j^k)$, which are P^k and $(\rho_j - P^k)$, respectively. Then, following the same token, we can reformulate (8) as the following two logically equivalent mixed-integer linear constraints:

$$y_i^k + \rho_j - y_j^k \;\; - P^k(1 - x_{i,j}^k) \le 0; \tag{20}$$

$$y_i^k + \rho_j - y_j^k \;\; - (1 - x_{i,j}^k)(\rho_j - P^k) \ge 0; \tag{21}$$

for all $i; j \in \mathcal{N}; i \neq j$. Finally, putting (16)–(17) and (20)–(21) together, we have the following logically-equivalent linear reformulation for Problem PMRS:

R-PMRS: Maximize $\underset{x}{\tilde{O}} \quad \tilde{O} \quad \tilde{O} \quad x_{i,j}^k u_{i,j}$

$k \in \mathcal{V}\; i \in \mathcal{N}\; j \in \mathcal{N}$

subject to Constraints (1)–(4), (16)–(17), (6)–(7), (20)–(21), (9)–(10):

Clearly, Problem R-PMRS is in the form of MILP thanks to the linearized constraints (16)–(17) and (20)–(21), so that it can be solved to optimality by various global optimization techniques (e.g., branch-and-cut, branch-and-bound, etc.) [3] when the problem size is moderate. Moreover, the global optimization solutions obtained by solving Problem R-PMRS provide a baseline for evaluating the exact and approximation algorithms developed in Sec. 5.

5 AN APPROXIMATION APPROACH WITH MEMORY-AUGMENTED TIME-EXPANSION

Although the MILP reformulation in Problem R-PMRS offers us global optimality guarantee, solving an MILP problem remains NP-Hard. Therefore, for large-sized problems, it remains necessary to develop efficient approximation algorithms. Toward this end, in this section, we introduce a memory-augmented time-expansion (MATE) approach, which will enable efficient approximation (or even exact) algorithm design. The rationale behind this approach is to *encode* time window feasibility, capacity and routing constraints into a MATE graph. By doing so, we transfer the structural complexity of the original PMRS problem into the MATE graph construction process, which turns out to be manageable in most ride-sharing settings. Toward this end, in Sec. 5.1, we first present the main idea and the overall algorithm of our MATE approach, which is followed by step-by-step illustrations of the MATE graph construction with a small example in Sec. 5.2. In Sec. 5.3, we provide complexity analysis for the MATE graph construction. Lastly, we state a new MATE-based problem reformulation in Sec. 5.4.

[3] These global optimization techniques are readily available in many existing optimization solvers, e.g., Gurobi, CPLEX, Mosek, etc.

5.1 The Main Idea and Overall Algorithm

The main idea of our MATE approach is to expand the original virtual graph in time domain, which allows us to encode the time window feasibility information into a new graph. Moreover, each node in the MATE graph is further augmented by rider group state information, which keeps track of the vehicle capacity constraints. Based on the new MATE graph, the original PMRS problem can be reformulated as a relatively well-structured multi-commodity flow problem without the complex time window feasibility and vehicle capacity constraints, which significantly simplifies the joint trip-vehicle matching and routing optimization.

Specifically, given a virtual graph $\mathcal{G} = (\mathcal{N}; \mathcal{L})$ as described in Sec. 3.1, we first expand $\mathcal{G}$ in time domain as $\mathcal{G}_{[\tau]} = (\mathcal{N}_{[\tau]}; \mathcal{L}_{[\tau]})$. Here, we let $\tau := \lambda_{\max} - \mu_{\min}$ denote the maximum number of time layers, where $\lambda_{\max} = \max_{v_a, r_i}\{\lambda_{d(v_a)}; \lambda_{d(r_i)}\}$ and $\mu_{\min} = \min_{v_b, r_j}\{\mu_{s(v_b)}; \mu_{s(r_j)}\}$. To simplify the notation in the new graph, we change the vehicle IDs by renaming $s(v_k)$ and $d(v_k)$ as s_k and d_k, respectively, $k = 0; \cdots; |\mathcal{V}| - 1$. Also, we change the rider group IDs by renaming $s(r_i)$ and $d(r_i)$ as $s_{|\mathcal{V}|+i}$ and $d_{|\mathcal{V}|+i}$, respectively, $i = 0; \cdots; |\mathcal{R}| - 1$. That is, the first $|\mathcal{V}|$ labels correspond to vehicles and remaining $|\mathcal{R}|$ labels correspond to rider groups.

The next key step is to augment each rider group node in the new time-expanded graph with memory to record a passing vehicle state (if any) , which further turns the time-expanded graph into a *state transition* graph under trip-vehicle matching and routing decisions. Toward this end, we define a new notation: at each source and destination node of a rider group i at time t, we let ω_t represent a passing vehicle state that records the set of rider groups on-board upon reaching this node. With ω_t, we label each rider group node in the MATE graph using the following format: $z_i(t; \omega_t)$, where $z_i \in \{s; d\}$ (i.e., z_i stands for either a source or destination node of rider group i) and t represents the time layer. As an example, the node in Fig. 2 corresponds to the source node of rider group 2 at time 3 with passing vehicle state $\{r_1; r_3\}$, that is, upon reaching this node, a passing vehicle has rider groups 1 and 3 on-board. We use $\{\cdot\}$ to represent undetermined passing vehicle state of a rider group source/destination. For the source/destination nodes of vehicles, we use $\{*\}$ to represent that their states must be empty. The construction of the MATE graph $\mathcal{G}_{[\tau]}$ is summarized as follow:

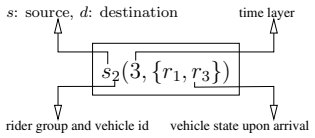

Figure 2: An illustration of rider group node labeling.

Algorithm 1: The MATE Graph Construction Algorithm.

Initialization:

1. Duplicate node set $\mathcal{N}$ into $(1 + \tau)$ layers, where each rider group node in each layer is labeled as $s_i(t; \{\cdot\})$ and $d_i(t; \{\cdot\})$, $t = 0; \cdots; \tau$, and each vehicle source/destination node in each layer is labeled as $s_i(t; \{*\})$ and $d_i(t; \{*\})$, $t = 0; \cdots; \tau$.

2. At each layer, remove nodes whose time indices are not within their departure time windows if they are sources or arrival windows if they are destinations. Let $t = 0$.

Main Loop:

3. In the t-th time layer. Replace all the unmodified rider group nodes $s_i(t;\{\cdot\})$ with $s_i(t;\{\emptyset\})$, and keep those rider group nodes who have already been updated in previous iterations.
4. For each node n in the current time layer, let i denote its corresponding rider group. Based on node n's passing vehicle state and routing constraints, look for all the next rider group hops that can be connected with time-feasible links (to be defined in Sec. 5.2).
5. For the next-hop node j, if it is a source node at time layer t' and its passing vehicle state $\omega_{t'}$ remains undetermined, we modify it as $s_j(t';\{\emptyset\})$. Expand this node j with updated passing vehicle state $\omega_t \cup \{r_i\}$ if node n is a source node, or $\omega_t \setminus \{r_i\}$ if node n is a destination node.
6. Create time-feasible links between each node n in time layer t with its next-hop nodes. Let $t = t + 1$ and go to Step 3 until $t > \tau$.

Finalization:

7. For every time layer, connect all drivers' source nodes with rider groups' source nodes that have empty vehicle states, i.e., $\omega_t = \{\emptyset\}$, using time-feasible links.
8. For every time layer, connect rider groups' destination nodes in the form of $d_i(t;\{r_i\})$ with all drivers' destination nodes using time-feasible links.
9. For every time layer, connect all drivers' source nodes with their corresponding destinations using time-feasible links.
10. For each vehicle i: for $t = \mu_{s_i};\cdots;\lambda_{s_i} - 1$, connect $s_i(t;\{*\})$ with $s_i(t+1;\{*\})$ using idle links (to be defined in Sec. 5.2); for $t = \mu_{d_i};\cdots;\lambda_{d_i} - 1$, connect $d_i(t;\{*\})$ with $d_i(t+1;\{*\})$ using idle links.

In the initialization stage of Algorithm 1, we expand the original virtual graph in time domain and remove nodes whose time indices clearly violate departure or arrival time windows of the corresponding rider groups or vehicles. In the main loop, we augment each node with memory to record all possible trip-vehicle matching and routing state transitions. In the finalization stage, we connect all vehicle nodes with expanded rider group nodes to represent all possible trip-vehicle matchings. To better illustrate the MATE graph construction process, in Sec. 5.2 we provide step-by-step details using a small illustrative example.

5.2 Step-by-step Illustration of the MATE Graph Construction

Now we demonstrate the details of graph construction in three stages as described in Algorithm-1. For illustration simplicity, we let vehicle capacity be two and the number of passengers in all the rider groups be one. We note, however, that the MATE graph construction process is applicable for general vehicle capacity and rider group size. Consider the virtual graph of a 3-rider-group and 1-vehicle example as shown in Fig. 3 and the corresponding vehicle and rider information provided in Table 2. In Fig. 3, the location of each node is defined by its coordinates. All distances in Table 2 is measured by Manhattan distance. As an example, the second row in

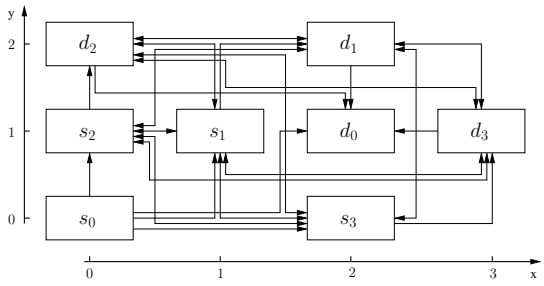

Figure 3: Relabeled virtual graph of a 3-rider-group and 1-vehicle example (ID 0: vehicle v_0; IDs 1 to 3: rider r_0 to r_2).

ID	s_i	d_i	$t_{s_i;d_i}$	μ_{s_i}	λ_{s_i}	μ_{d_i}	λ_{d_i}
$1(r_1)$	(1,1)	(2,2)	2	2	2	4	4
$2(r_2)$	(0,1)	(0,2)	1	1	2	2	3
$3(r_3)$	(2,0)	(3,1)	2	2	4	4	6
$0(v_0)$	(0,0)	(2,1)	3	0	2	3	5

Table 2: Rider and vehicle information of the example in Fig. 3.

Table 2 means that rider group r_2 is labeled as ID 2; its source and destination are located at (0; 1) and (0; 2), respectively; the direct travel time between them is 1 (assuming the travel speed is one distance unit per time unit). Suppose μ_{s_2} and λ_{d_2} are given as in Table 2, following the time window constraints in Sec. 3.2, we can compute that $\lambda_{s_2} = 3 - 1 = 2$ and $\mu_{d_2} = 1 + 1 = 2$.

Initialization (Steps 1-2): Given the data in Table 2, it follows that $\tau = 6 - 0 = 6$. Thus, we duplicate the node set into $6 + 1$ time layers as shown in Fig. 4a (due to space limitation, we only show three time layers with $t = 0; 1; 2$). Next, we remove nodes whose time windows are violated, i.e., for a rider or vehicle ID i, the source nodes whose time indices not in the departure time window $[\mu_{s_i};\lambda_{d_i} - t_{s_i;d_i}]$ are removed, the destination nodes whose time indices not in the arrival time window $[\mu_{s_i} + t_{s_i;d_i};\lambda_{d_i}]$ are removed as well. For example, since the departure time window of s_2 is $[1; 2]$ (see the computation above), node s_2 at layer 0 is removed in Fig. 4a (marked as a white box). For easier visualization of all time layers, we map the 3-D graph in Fig. 4a into a 2-D graph with labels in the format of $z_i(t;\{\cdot\})$ as shown in Fig. 4b. For example, nodes s_0 and s_2 in layer $t = 1$ are mapped to the two nodes $s_0(1;\{*\})$ and $s_2(1;\{\cdot\})$, respectively.

Main Loop (Steps 3-6): Since there is no rider group node in layer $t = 0$, we consider layer $t = 1$. Following Algorithm-1, we initialize unmodified rider nodes whose vehicle state ω_t is $\{\cdot\}$. Hence, $s_2(1;\{\cdot\})$ is replaced with $s_2(1;\{\emptyset\})$, which is already done in Fig. 5a (marked as bold). We now formally define the notion of *time-feasible link* as mentioned in Step 4 of Algorithm-1:

Definition 5.1 (Time-feasible Link). Let $z_i(t;\omega_t)$ be a source or destination node in $\mathcal{N}_{[\tau]}$. Given another node $z_j(t';\omega_{t'}) \in \mathcal{N}_{[\tau]}$, a directed link from $z_i(t;\omega_t)$ to $z_j(t';\omega_{t'})$ is called a time-feasible link if the following conditions are satisfied:

- A directed edge $(z_i; z_j)$ exists in the virtual graph $\mathcal{G}$;
- Time t falls in the departure time window of node z_i: $t \in [\mu_{z_i};\lambda_{z_i}]$;
- Time $t' = t + t_{i;j}$ is in the arrival time window at z_j: $t' \in [\mu_{z_j};\lambda_{z_j}]$.

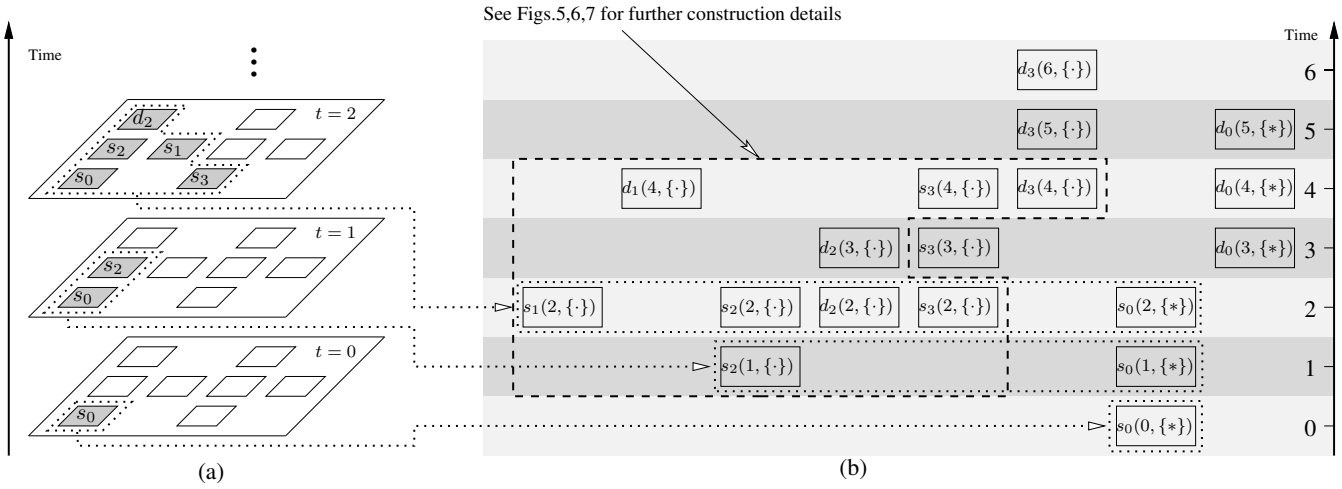

Figure 4: Time-expanded, remove nodes where time windows are not satisfied

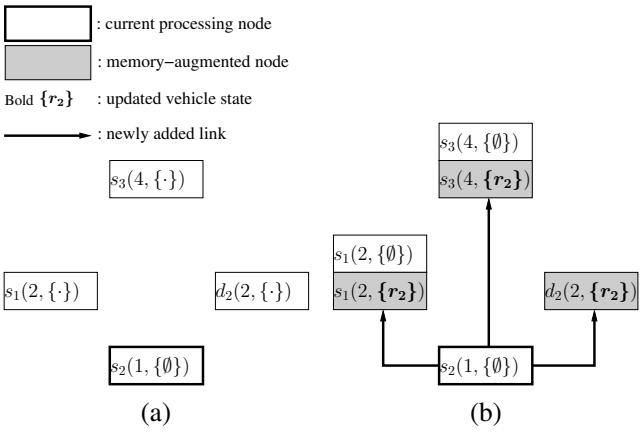

Figure 5: Processing nodes of time layer $t = 1$: $s_2(1;\{\emptyset\})$.

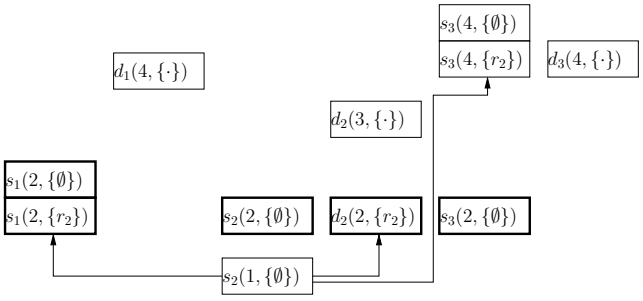

Figure 6: Processing nodes of time layer $t = 2$.

With the notion of time-feasible link, the construction of MATE graph proceeds as follows: As mentioned in Step 4 in Algorithm 1, we look for next time-feasible hops of $s_2(1;\{\emptyset\})$. The vehicle picks up rider group r_2 at this node, and based on its passing vehicle state $\{\emptyset\}$ and routing constraints, it can go to any other new rider group sources or destination node of r_2, i.e., $s_1(2;\{\cdot\})$, $s_3(4;\{\cdot\})$ and $d_2(2;\{\cdot\})$ as shown in Fig. 5a. Since $s_2(1;\{\emptyset\})$ has no rider group

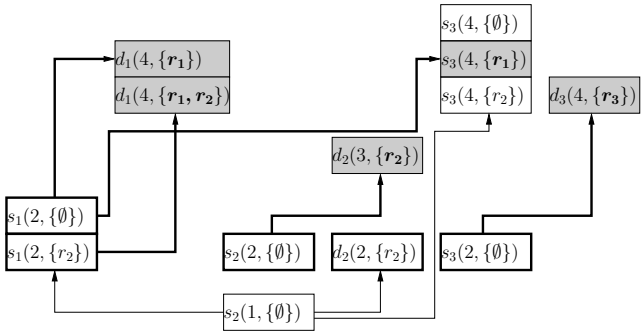

Figure 7: Expand next-hop nodes and make them connected.

on-board, after leaving this node, the next-hop vehicle state is $\{r_2\}$ (see shaded boxes in Fig. 5b). Following Step 5, we create $s_1(2;\{\emptyset\})$ and $s_3(4;\{\emptyset\})$ for $s_1(2;\{\cdot\})$ and $s_3(4;\{\cdot\})$, respectively, then expand these three nodes with updated vehicle state $\{r_2\}$. Then, following Step 6, we create time-feasible links from $s_2(1;\{\emptyset\})$ to the expanded nodes as shown in Fig. 5b (shaded boxes). The processing for the time layer $t = 1$ is completed. [4]

In the next iteration, i.e., time layer $t = 2$, following Step 3, among all the rider nodes, we initialize unmodified rider nodes whose vehicle state ω is $\{\cdot\}$, others remain the same. Hence, $s_2(2;\{\cdot\})$ and $s_3(2;\{\cdot\})$ are replaced with $s_2(2;\{\emptyset\})$ and $s_3(2;\{\emptyset\})$, respectively, which is already done in Fig. 6. As mentioned in Steps 4–6 of Algorithm 1, we look for the next time-feasible hops of $s_1(2;\{\emptyset\})$, $s_1(2;\{r_2\})$, $s_2(2;\{\emptyset\})$, $d_2(2;\{r_2\})$ and $s_3(2;\{\emptyset\})$ (marked as bold in Fig. 6. The following process is illustrated in Fig. 7:

- Next time-feasible hops of $s_1(2;\{\emptyset\})$ are $d_1(4;\{\cdot\})$ and $s_3(4;\{\emptyset\})$. Next-hop vehicle state is $\{r_1\}$. We modify $d_1(4;\{\cdot\})$ as $d_1(4;\{r_1\})$. Since $s_3(4;\emptyset)$ already exists, we expand it into $s_3(4;\{r_1\})$ and $s_3(4;\emptyset)$. Connect them with time-feasible links from $s_1(2;\{\emptyset\})$;

[4] Notice that $s_2(2;\{\emptyset\})$ and $s_2(2;\{r_2\})$ have the same rider source s_2 and the same time layer $t = 2$, but different vehicle states. Thus, they are two different nodes although being stacked together for space saving and clear illustration. Note that there is no link between them.

- Next time-feasible hops of $s_1(2;\{r_2\})$ is d_1 only. Note that although r_2 is on-board, the vehicle cannot go to d_2 at time layer $t = 2$ since it takes two units of time from s_2 to d_2 (see Fig. 3) but there is no d_2-node at time layer $t = 4$ (see Fig. 4b). After passing $s_1(2;\{r_2\})$, the vehicle state becomes $\{r_1; r_2\}$. Hence, we can only expand $d_1(4;\{r_1\})$ into $d_1(4;\{r_1; r_2\})$ and $d_1(4;\{r_1\})$. Connect $s_1(2;\{r_2\})$ with $d_1(4;\{r_1; r_2\})$;
- Next time-feasible hop of $s_2(2;\{\emptyset\})$ is $d_2(3;\{\cdot\})$. We modify it as $d_2(3;\{r_2\})$ and connect it with a time feasible link from $s_2(2;\{\emptyset\})$;
- Next time-feasible hop of $d_2(2;\{r_2\})$ can be a new rider source, since r_2 is dropped off at this node, i.e., the next-hop vehicle state is $\{\emptyset\}$. If there is a source s_i at another time layer t' for which $t + t_{d_2 s_i} = t'$, a time-feasible link from $d_2(2;\{r_2\})$ to that s_i-node can be added. In this example, however, no such nodes exist;
- Next time-feasible hop of $s_3(2;\{\emptyset\})$ is $d_3(4;\{\cdot\})$. We modify it as $d_3(4;\{r_3\})$ and connect it with a time feasible link from $s_3(2;\{\emptyset\})$.

Similar process will be repeated at other time layers until $t > \tau$. We omit the details of the rest of the main loop for brevity.

Finalization (Steps 7-9): We connect rider nodes with vehicle nodes (see Fig. 8). Toward this end, we need the following definition:

Definition 5.2 (Vehicle Idle Link). Let $z_i(t;\{*\})$ be a source or destination node of vehicle i, the link connecting $z_i(t;\{*\})$ and $z_i(t+1;\{*\})$ is a vehicle idle link (with zero distance and profit).

With the notion of vehicle idle link, the finalization stage proceeds as follows (see bold links in Fig. 8):

- For time layer $t = 0; 1; 2$, connect $s_0(t;\{*\})$ with rider groups' source nodes that have empty vehicle states ($s_1(2;\{\emptyset\}), s_2(1;\{\emptyset\})$, $s_2(2;\{\emptyset\}), s_3(2;\{\emptyset\})$ and $s_3(4;\{\emptyset\})$) using time-feasible links.
- For time layer $t = 3; 4; 5$, connect rider groups' destination nodes in the form of $d_i(t;\{r_i\})$ ($d_1(4;\{r_1\}), d_2(2;\{r_2\})$ and $d_3(4;\{r_3\})$) with all drivers' destination nodes using time-feasible links.
- For time layer $t = 0; 1; 2$, connect the vehicle's source node with its corresponding destination using a time-feasible link.
- For time layer $t \in [0; 5]$, add vehicle idle links for vehicle's source nodes and destination nodes.

The completed MATE graph for this example is shown in Fig. 8.

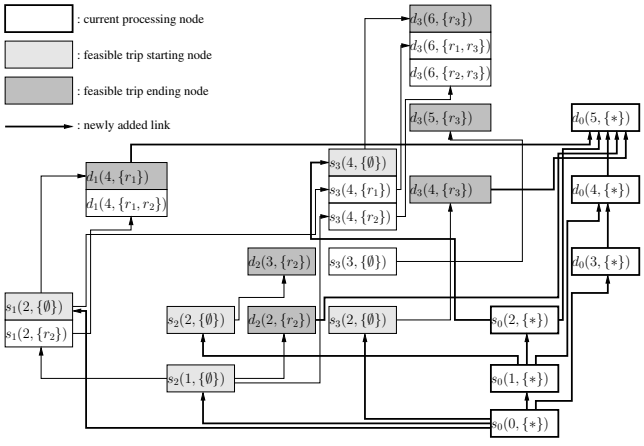

Figure 8: Connect rider group nodes with vehicle nodes. A completed MATE graph

5.3 Complexity of MATE Graph Construction

In this section, we analyze the complexity of the MATE graph construction. Let $\Omega(i;t)$ be the set of expanded nodes that share the same rider group node and time information but have different vehicle states (see, e.g., $s_1(2;\{\emptyset\})$ and $s_1(2;\{r_2\})$ in Fig. 8). Let R be the number of rider groups, V be vehicle number, and K be vehicle capacity. We assume $R \geq K$ and $R \geq V$ to avoid triviality.

Lemma 5.3. *Given R riders and vehicles capacity K in $\mathcal{G}_{[\tau]}$, the total number of nodes is of order $O(\tau R(\sum_{i=0}^{K-1}\binom{R-1}{i}))$, and the total number of links is of order $O(\tau R(\sum_{i=0}^{K-1}\binom{R-1}{i})(R-1))$.*

Proof. Since there are at most $2\tau R$ such $\Omega(i;t)$ node sets and $2\tau V$ vehicle nodes, the total number of nodes can be computed as $2\tau R(\sum_{i=0}^{K-1}\binom{R-1}{i}) + 2\tau V = O(\tau R(\sum_{i=0}^{K-1}\binom{R-1}{i}))$. Since there are at most $(R-1)$ outgoing links for each rider node, the total number of links among rider nodes is at most $2\tau R(\sum_{i=0}^{K-1}\binom{R-1}{i})(R-1)$. The total number of links connected with vehicle nodes is at most $2\tau RV + \tau V$. Therefore, the total number of links is $2\tau R(\sum_{i=0}^{K-1}\binom{R-1}{i})(R-1) + (2R+1)\tau V = O(\tau R(\sum_{i=0}^{K-1}\binom{R-1}{i})(R-1))$.

Based on Lemma 5.3, we can show the following result on the time and memory complexity of the MATE graph construction.

Theorem 5.4 (Time and Memory Complexity of MATE Graph). *Given R rider groups and V vehicles of capacity K, the time and memory complexities of Algorithm 1 are both $O(\tau R^2((R-1)^{K-1}+1))$.*

Proof. Recall that maximum size of $\Omega(i;t)$ is $\binom{K-1}{i=0}\binom{R-1}{i}$ and $K \leq R$. Now, we claim that $\sum_{i=0}^{K-1}\binom{R-1}{i} \leq (R-1)^{K-1}+1$, which can be proved by induction on the value of R. For the base case $R = 2$ (i.e., $R - 1 = 1$), it follows that $K = 0; 1; 2$. If $K = 0$, $\sum_{i=0}^{K-1}\binom{R-1}{i}$ is vacuous. If $K = 1$, $\sum_{i=0}^{K-1}\binom{R-1}{i} = 0 \leq 1+1$. If $K = 2$, $\sum_{i=0}^{K-1}\binom{R-1}{i} = 1 + R - 1 \leq R - 1 + 1$. Thus, the base case is proved.

Suppose that the hypothesis is true for $R-1$. Next, we show that the hypothesis continues to hold for R. If $K - 1 = R$, then it is clear that $\sum_{i=0}^{K-1}\binom{R}{i} = \sum_{i=0}^{R}\binom{R}{i}1^i 1^{R-i} = (1+1)^R = 2^R \leq R^R + 1$. If $K = 0$, the hypothesis trivially holds: $\binom{R}{0} = 1 \leq R^0 + 1$. For $1 \leq K - 1 \leq R - 1$, we have the following derivation:

$$\begin{aligned}
\sum_{i=0}^{K-1}\binom{R}{i} &= 1 + \sum_{i=1}^{K-1}\binom{R}{i} \overset{(a)}{=} 1 + \sum_{i=1}^{K-1}\binom{R-1}{i} + \binom{R-1}{i-1} \\
&= \sum_{i=0}^{K-1}\binom{R-1}{i} + \sum_{i=1}^{K-1}\binom{R-1}{i-1} = \sum_{i=0}^{K-1}\binom{R-1}{i} + \sum_{j=0}^{K-2}\binom{R-1}{j} \\
&\overset{(b)}{\leq} (R-1)^{K-1} + 1 + (R-1)^{K-2} + 1 \\
&\leq (R-1)^{K-1} + (K-1)(R-1)^{K-2} + 1 + 1 \overset{(c)}{\leq} R^{K-1} + 1; \qquad (22)
\end{aligned}$$

where (a) follows from the Pascal's formula $\binom{R}{i} = \binom{R-1}{i} + \binom{R-1}{i-1}$; (b) follows from the induction hypothesis; and (c) follows from the binomial expansion $R^{K-1} = (1+R-1)^{K-1} = (R-1)^{K-1} + (K-1)(R-1)^{K-2} + \cdots + 1 \geq (R-1)^{K-1} + (K-1)(R-1)^{K-2} + 1$. Hence, the claim is proved. Note that each node and link operation needs the same amount of time and memory. Then, the stated complexity result $O(\tau R^2((R-1)^{K-1}+1))$ follows from Lemma 5.3 and (22).

Theorem 5.4 suggests that the complexity of MATE graph construction is polynomial in K. Note that vehicle capacity K is usually small in ride-sharing (e.g., 4 seats). Thus, the complexity of MATE approach is manageable in most ride-sharing settings.

5.4 The MATE-based Problem Reformulation

Now that we have MATE graph $\mathcal{G}_{[\tau]}$ constructed, we will develop a reformulated problem based on the MATE graph. First, we note that since time window feasibility, capacity and routing constraints are already encoded in the MATE graph, the MATE-based reformulation no longer contains constraints (5) – (10). As a result, the only surviving constraints are vehicle routing constraints (1) – (4), which are in the multi-commodity flow-balance form and relatively easy to handle. Next, we will translate these vehicle routing constraints into a set of new multi-commodity flow balance constraints in the MATE graph. Note that the source of a commodity corresponding to a vehicle is the vehicle's source node at its earliest departure time layer. Likewise, the sink of a commodity corresponding to a vehicle is the vehicle's destination node at its latest arrival time layer. For example, in Fig. 8, the source and sink nodes of the vehicle are $s_0(0;\{*\})$ and $d_0(5;\{*\})$, respectively. For notation simplicity, we denote the source and destination nodes of the i-th commodity (i.e., vehicle) in the MATE graph as S_i and D_i, respectively. Next, to model the fact that a rider group cannot be picked up by more than one vehicle in all time layers in the MATE graph, we transform the constraint in (2) as follows:

$$\sum_{k\in\mathcal{V}}\sum_{t=0}^{}\sum_{i\in\Omega(\mathsf{s}_r;t)}\sum_{j;l_{i;j}\in\mathcal{L}_{[\tau]}} x^k_{i;j} \le 1;\forall r\in\mathcal{R}: \tag{23}$$

Note that the indices i and j in the binary variable $x^k_{i;j}$ now correspond to nodes in the MATE graph rather than the original virtual graph. With this distinction, flow-balance constraints (1), (2) and (4) based on the original virtual graph should also be changed based on the MATE graph as shown in the following reformulation:

MATE-PMRS:

$$\underset{x}{\text{Maximize}} \sum_{k\in\mathcal{V}}\sum_{(i;j)\in\mathcal{L}_{[\tau]}} x^k_{i;j}u_{i;j} \tag{24}$$

$$\text{subject to} \sum_{i;(\mathsf{S}_k;i)\in\mathcal{L}_{[\tau]}} x^k_{\mathsf{S}_k;i} = 1; \sum_{j;(j;\mathsf{D}_k)\in\mathcal{L}_{[\tau]}} x^k_{j;\mathsf{D}_k} = 1;\forall k\in\mathcal{V}; \tag{25}$$

$$\sum_{i;(i;l)\in\mathcal{L}_{[\tau]}} x^k_{i;l} = \sum_{j;(l;j)\in\mathcal{L}_{[\tau]}} x^k_{l;j};\forall l\in\mathcal{N}_{[\tau]}\setminus\{\mathsf{S}_k;\mathsf{D}_k\};\forall k\in\mathcal{V}; \tag{26}$$

$$x^k_{i;j}\in\{0;1\}; x^k_{i;j}=0, \text{ if no such link is in } \mathcal{G}_{[\tau]}; \tag{27}$$

and Constraint (23):

In Problem MATE-PMRS, the first three sets of constraints are the new multi-commodity flow-balance constraints and the fourth constraint is exactly (23). Although Problem MATE-PMRS remains an MILP, it has well-structured flow-balance-like constraints that are close to a multi-commodity flow problem over the MATE graph. This enables us to develop efficient approximation (or even exact) algorithms to solve the problem in the next section.

6 SOLVING THE MATE-BASED PROBLEM

In this section, we will develop efficient algorithms for solving Problem MATE-PMRS. In Sec. 6.1, we first consider the single-vehicle setting, where the MATE approach entails optimal solutions due to the underlying total unimodularity (TU) in this setting. We then consider the multi-vehicle setting in Sec. 6.2, where the MATE approach enables the design of approximation algorithms.

6.1 Single-vehicle Setting

We first consider the single-vehicle setting, which is also interesting in its own right. This is because when the time unit is sufficiently small, the probability of having more than one vehicle to match requesting rider groups is small. Moreover, the single-vehicle setting naturally fits into an online setting, where a stream of vehicles arrive one by one sequentially. However, the ride-sharing service provider does not have future arrival information. Thus, trip-vehicle matching and routing decisions have to be made one at a time. In the single-vehicle setting, Problem MATE-PMRS reduces to:

MATE-PMRS-SINGLE:

$$\underset{x}{\text{Maximize}} \sum_{(i;j)\in\mathcal{L}_{[\tau]}} x_{i;j}u_{i;j} \tag{28}$$

$$\text{subject to} \sum_{i;(\mathsf{S};i)\in\mathcal{L}_{[\tau]}} x_{\mathsf{S};i} = 1; \sum_{j;(j;\mathsf{D})\in\mathcal{L}_{[\tau]}} x_{j;\mathsf{D}} = 1; \tag{29}$$

$$\sum_{i;(i;u)\in\mathcal{L}_{[\tau]}} x_{i;u} = \sum_{j;(u;j)\in\mathcal{L}_{[\tau]}} x_{u;j};\forall u\in\mathcal{N}_{[\tau]}\setminus\{\mathsf{S};\mathsf{D}\}; \tag{30}$$

$$\sum_{t=0}^{}\sum_{i\in\Omega(\mathsf{s}_r;t)}\sum_{j;l_{i;j}\in\mathcal{L}_{[\tau]}} x_{i;j} \le 1;\forall r\in\mathcal{R}; \tag{31}$$

$$x_{i;j}\in\{0;1\}; x_{i;j}=0, \text{ if no such link in } \mathcal{G}_{[\tau]}: \tag{32}$$

Note that if Constraint (31) is removed, Problem MATE-PMRS-SINGLE can be further reduced to a nice *single-commodity network flow* problem, which is well-known to be polynomially solvable thanks to its TU property. Interestingly, Constraint (31) can indeed be removed in practical cases where the assumption below holds:

Assumption 1. *Time horizon τ does not exceed twice of the direct travel time of any rider group, i.e., $\tau < 2t_{s_i;d_i};\forall i\in\mathcal{R}$.*

We note that Assumption 1 holds in practice because, according to the survey [23], riders are highly unlikely to opt in UberPool when the potential trip duration of ride-sharing is more than 1.4 times of the direct travel time. Then, we have the following result:

Proposition 6.1. *Under Assumption 1, any rider group can be served no more than once in the single-vehicle setting. Therefore, Constraint* (31) *is satisfied automatically and can be removed.*

Now we can safely remove Constraint (31) of Problem MATE-PMRS-SIN and obtain a new formulation as follows:

MATE-PMRS-SINGLE-R:

$\underset{x}{\text{Maximize}}$ (28), subject to Constraints (29), (30) and (32):

Note that Problem MATE-PMRS-SIN-R is a single-commodity network flow problem. Although this problem is still an integer program (IP), it can be readily verified that the constraints in (29), (30) and (32) satisfy the TU property defined as follows:

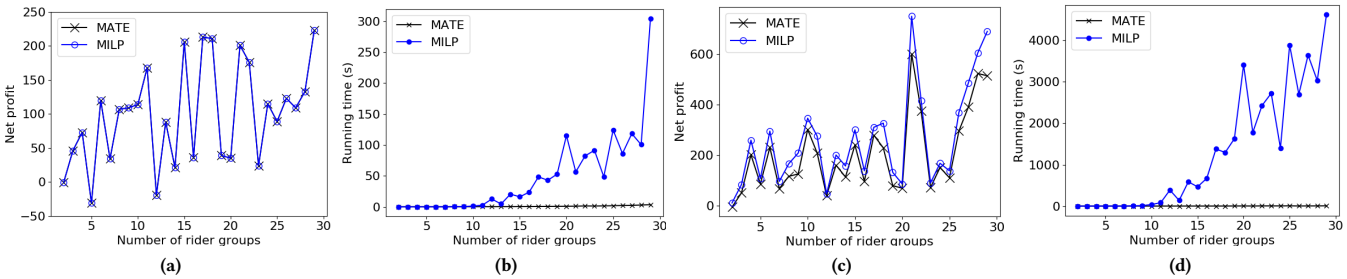

Figure 9: (a)–(b): Profit and running time in single-vehicle setting; (c)–(d): profit and running time in multi-vehicle setting.

Definition 6.2 (Total Unimodularity). A matrix $\mathbf{A}$ is totally unimodular if each square submatrix of $\mathbf{A}$ has determinant ± 1 or 0.

Furthermore, the TU property implies the following result:

Proposition 6.3 (Integral Optimality of TU [29]). *Consider the integer programming problem (IP):* $\max \mathbf{u}^\top \mathbf{x}$, *s.t.* $\mathbf{A}\mathbf{x} = \mathbf{b}; \mathbf{x} \in \mathbb{Z}^n$, *where* $\mathbf{A} \in \mathbb{Z}^{m\times n}$ *and* $\mathbf{b} \in \mathbb{Z}^m$ *have integer coefficients, and* $\mathbf{A}$ *is TU, then (IP) can be relaxed as a linear program (LP). (LP) solves (IP) for all* $\mathbf{b} \in \mathbb{Z}^m$ *for which an optimal solution exists.*

Based on Proposition 6.3, we can solve MATE-PMRS-SINGLE-R as an LP in polynomial time, and the optimal solution is integral.

6.2 Multiple-vehicle Setting

In the multi-vehicle setting, even with Assumption 1, Constraint (23) in Problem MATE-PMRS still cannot be removed because Assumption 1 does not imply that a rider group can be served by only one vehicle (it only implies a rider group can be served by the same vehicle only once). Therefore, Problem MATE-PMRS cannot be further reduced to a multi-commodity network flow problem. To address this challenge, we consider a penalized version of Problem MATE-PMRS as follows:

MATE-PMRS-P:

$$\underset{x}{\text{Maximize}} \quad (24) - \sum_{r\in\mathcal{R}} w_r \sum_{k\in\mathcal{V}} \sum_{t=0} \sum_{i\in\Omega(s_r,t)} \sum_{j,l_{i,j}\in\mathcal{L}_{[\tau]}} x^k_{i,j} - 1$$

subject to Constraints (25), (26) and (27);

where parameters $w_r > 0; \forall r \in \mathcal{R}$ are large to heavily penalize the violation of Constraint (23). As $w_r \to \infty$, Problem MATE-PMRS-P closely approximates Problem MATE-PMRS. Note that Problem MATE-PMRS-P is a multi-commodity network flow problem. Hence, any state-of-the-art approximation algorithm for solving multi-commodity flow problem [10, 12, 19, 40] can be readily applied.

7 NUMERICAL RESULTS

In this section, we conduct numerical experiments to verify the efficacy of our MATE approach. We randomly sample 25,000 taxi trip records of Jan 1st 2016 from the New York City Taxi dataset [31], which contains pick-up and drop-off locations and departure/arrival time information. Average speed is chosen as 15Mph according to a mobility report from NYC Department of Transportation [30]. Drivers' cost coefficients are all chosen as $\alpha = 1$ per mile. Riders' payment coefficients are all set to $\beta = 4$ per mile.

Single-Vehicle Setting: We let each rider group have one passenger and vary the number of rider groups from 2 to 29. We then randomly select one more trip to play the role of the vehicle driver. We run both MILP and MATE approaches. The net profit and running time comparisons are illustrated in Fig. 9a and Fig. 9b, respectively. We can see from Fig. 9a that the net profits obtained by MILP and MATE are exactly equal, which confirms that the TU property implied by the MATE approach under the single-vehicle setting leads to *global optimal* solution (always achievable by MILP). However, as the number of rider groups increases, the running time of the MILP approach increases roughly exponentially, while the running time of MATE is insensitive to the rider group numbers.

Multi-Vehicle Setting: We adopt the classic FPTAS (fully polynomial time approximation scheme) algorithm for multi-commodity network flow [10] to solve Problem MATE-PMRS-P with two vehicles (approximation tolerance $\epsilon = 0.15$) and compare the results to those obtained by solving Problem R-PMRS with the MILP approach, and the results are illustrated in Fig. 9c and Fig. 9d. We can see the MATE approach closely approximates the global optimal solutions obtained by the MILP approach (the results nearly coincide in some cases in Fig. 9c). We again observe similar trends of running time in Fig. 9d, where the MILP approach slows down significantly as the number of rider group increases, while the running time of the MATE approach increases slowly.

8 CONCLUSION

In this paper, we studied the problem of joint vehicle-trip matching and routing optimization in ride-sharing systems. We first proposed a new analytical framework and showed that the problem can be formulated as an NP-Hard mixed-integer nonlinear programming problem. To overcome this challenge, we proposed two approaches: i) reformulating the problem as a mixed-integer linear program, for which moderate-sized instances can be solved by global optimization methods much more easily; and ii) a memory-augmented time-expansion (MATE) approach, which exploits the special graphical structure of the problem to enable approximate (or even exact) algorithm designs. Extensive numerical experiments are conducted to verify the proposed approaches. We note that joint trip-vehicle matching and routing optimization is an under-explored problem lying at the heart of ride-sharing systems. Future directions may include more sophisticated models that incorporate uncertainties in riders' arrivals and vehicles' speeds, heterogeneous commute patterns, and their associated algorithm designs.

REFERENCES

[1] 2020. IBM ILOG CPLEX optimizer. http://www.cplex.com

[2] J. Alonso-Mora, S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus. 2017. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences* 114, 3 (2017), 462–467. https://doi.org/10.1073/pnas.1611675114

[3] MOSEK ApS. 2019. *The MOSEK optimization toolbox for MATLAB manual. Version 9.0.* http://docs.mosek.com/9.0/toolbox/index.html

[4] A. Atahran, C. Lenté, and V. T'kindt. 2014. A Multicriteria Dial-a-Ride Problem with an Ecological Measure and Heterogeneous Vehicles. *Journal of Multi-Criteria Decision Analysis* 21 (2014). https://doi.org/10.1002/mcda.1518

[5] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali. 2010. *Linear Programming and Network Flows* (4th ed.). John Wiley and Sons, New York.

[6] X. Bei and S. Zhang. 2017. Algorithms for Trip-Vehicle Assignment in Ride-Sharing. (2017), 3–9. http://www.ntu.edu.sg/home/xhbei/papers/ridesharing.pdf

[7] G. Berbeglia, J. Cordeau, and G. Laporte. 2011. A Hybrid Tabu Search and Constraint Programming Algorithm for the Dynamic Dial-a-Ride Problem. *INFORMS Journal on Computing* 24, 3 (May 2011). https://doi.org/10.1287/ijoc.1110.0454

[8] K. Braekers and A. A. Kovacs. 2016. A multi-period dial-a-ride problem with driver consistency. *Transportation Research Part B: Methodological* 94 (2016), 355–377. https://doi.org/10.1016/j.trb.2016.09.010

[9] J. Cordeau and G. Laporte. 2007. The dial-a-ride problem: models and algorithms. *Annals of Operations Research* 153 (May 2007), 29–46. https://doi.org/10.1007/s10479-007-0170-8

[10] L. K. Fleischer. 1999. Approximating fractional multicommodity flow independent of the number of commodities. *Annual Symposium on Foundations of Computer Science - Proceedings* 13, 4 (1999), 24–31.

[11] T. Garaix, C. Artigues, D. Feillet, and D. Josselin. 2011. Optimization of occupancy rate in dial-a-ride problems via linear fractional column generation. *Computers & Operations Research* 38 (2011), 1435–1442. https://doi.org/10.1016/j.cor.2010.12.014

[12] N. Garg and J. K. Onemann. 2007. Faster and Simpler Algorithms for Multicommodity Flow and Other Fractional Packing Problems. 37, 2 (2007), 630–652.

[13] LLC Gurobi Optimization. 2019. Gurobi Optimizer Reference Manual. http://www.gurobi.com

[14] G. Heilporn, J. Cordeau, and G. Laporte. 2011. An integer L-shaped algorithm for the Dial-a-Ride Problem with stochastic customer delays. *Discrete Applied Mathematics* 159 (Jun. 2011), 883–895. https://doi.org/10.1016/j.dam.2011.01.021

[15] W. Herbawi and M. Weber. 2012. A genetic and insertion heuristic algorithm for solving the dynamic ridematching problem with time windows. *Ieee Wcci* (2012), 385–392. https://doi.org/10.1109/CEC.2012.6253001

[16] S. Jain and P. Van Hentenryck. 2011. Large Neighborhood Search for Dial-a-Ride Problems Large Neighborhood Search for Dial-a-Ride Problems. *International Conference on Principles and Practice of Constraint Programming* (2011), 400–413. https://doi.org/10.1007/978-3-642-23786-7_31

[17] D. Kirchler and R. Calvob. 2013. A Granular Tabu Search algorithm for the Dial-a-Ride Problem. *Transportation Research Part B: Methodological* 56 (Oct. 2013), 120–135. https://doi.org/10.1016/j.trb.2013.07.014

[18] F. Lehuédé, R. Masson, S. N. Parragh, O. Péton, and F. Tricoire. 2014. A multi-criteria large neighbourhood search for the transportation of disabled people. *Journal of the Operational Research Society* 65, 7 (2014), 983–1000. https://doi.org/10.1057/jors.2013.17

[19] T. Leighton, F. Makedont, S. Plotkin, C. Steins, E. Tardos, and S. Tragoudas. 1991. Fast approximation algorithms for multicommodity flow problems. *Proceedings of the Annual ACM Symposium on Theory of Computing* Part F130073 (1991), 101–111. https://doi.org/10.1145/103418.103425

[20] Q. Lin, L. Deng, J. Sun, and M. Chen. 2018. Optimal Demand-Aware Ride-Sharing Routing. *International Conference on Computer Communications (INFOCOM)* ii (2018), 1–13.

[21] Q. Lin, W. Xu, M. Chen, and X. Lin. 2019. A Probabilistic Approach for Demand-Aware Ride-Sharing Optimization. July (2019).

[22] M. Liu, Z. Luo, and A .Lim. 2015. A branch-and-cut algorithm for a realistic dial-a-ride problem. *Transportation Research Part B: Methodological* 81 (2015), 267–288. https://doi.org/10.1016/j.trb.2015.05.009

[23] J. Lo and S. Morseman. 2018. The Perfect uberPOOL: A Case Study on Trade-Offs. *Ethnographic Praxis in Industry Conference Proceedings* 2018, 1 (2018), 195–223. https://doi.org/10.1111/1559-8918.2018.01204

[24] M. A. Masmoudi, M. Hosny amd K. Braekers, and A. Dammak. 2016. Three effective metaheuristics to solve the multi-depot multi-trip heterogeneous dial-a-ride problem. *Transportation Research Part E: Logistics and Transportation Review* 96 (2016), 60–80. https://doi.org/10.1016/j.tre.2016.10.002

[25] M. A. Masmoudi, K. Braekers, M. Masmoudi, and A. Dammak. 2017. A hybrid Genetic Algorithm for the Heterogeneous Dial-A-Ride Problem. *Computers & Operations Research* 81 (2017), 1–13. https://doi.org/10.1016/j.cor.2016.12.008

[26] R. Masson, F. Lehuede, and O. Peton. 2014. The Dial-A-Ride Problem with Transfers. *Computers & Operations Research* 41 (2014), 12–23. https://doi.org/10.1016/j.cor.2013.07.020

[27] J. Miller and J. P. How. 2017. Predictive positioning and quality of service ridesharing for campus mobility on demand systems. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 1402–1408. https://doi.org/10.1109/ICRA.2017.7989167

[28] A. Mourad, J. Puchinger, and C. Chu. 2019. A survey of models and algorithms for optimizing shared mobility. *Transportation Research Part B: Methodological* 123 (2019), 323–346. https://doi.org/10.1016/j.trb.2019.02.003

[29] G. Nemhauser and L. Wolsey. 1988. *Integer and Combinatorial Optimization*. John Wiley and Sons.

[30] NYCDOT. 2019. Mobility report from NYC Department of Transportation. https://www1.nyc.gov/html/dot/downloads/pdf/mobility-report-2019-print.pdf

[31] NYCTLC. 2019. NYC Taxi TLC Trip Record Data. https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

[32] S. Parragh and V. Schmid. 2013. Hybrid column generation and large neighborhood search for the dial-a-ride problem. *Computers & Operations Research* 40 (Jan. 2013), 490–497.

[33] S. N. Parragh, J. P. Sousa, and B. Almada-Lobo. 2015. The Dial-a-Ride Problem with Split Requests and Profits. *Transportation Science* 49, 2 (2015). https://doi.org/10.1287/trsc.2014.0520

[34] Partnership for New York City 2018. *$100 Billion Cost of Traffic Congestion in Metro New York. Internet:*. Partnership for New York City. http://pfnyc.org/wp-content/uploads/2018/01/2018-01-Congestion-Pricing.pdf

[35] V. Pimenta, A. Quilliot, H. Toussaint, and D. Vigo. 2017. Models and algorithms for reliability-oriented Dial-a-Ride with autonomous electric vehicles. *European Journal of Operational Research* 257 (2017), 601–613. https://doi.org/10.1016/j.ejor.2016.07.037

[36] U. Ritzinger, J. Puchinger, and R. F. Hartl. 2016. A survey on dynamic and stochastic vehicle routing problems. *International Journal of Production Research* 54 (2016), 215–231. https://doi.org/10.1080/00207543.2015.1043403

[37] S. Røpke, J. Cordeau, and G. Laporte. 2007. Models and a Branch-and-Cut Algorithm for Pickup and Delivery Problems with Time Windows. *Networks* 49, 43 (2007), 258–272. https://doi.org/10.1002/net.20177

[38] J. Saranow. 2006. Carpooling for grown-ups high gas prices, new services give ride-sharing a boost, rating your fellow rider. https://www.wsj.com/articles/SB113884611734062840

[39] H. D. Sherali and W. P. Adams. 1999. *A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems*. Kluwer Academic Publishing, Boston, MA.

[40] I. Wang. 2018. Multicommodity Network Flows : A Survey , Part I : Applications and Formulations. *International Journal of Operations Research* 15, 4 (2018), 145–153. https://doi.org/10.6886/IJOR.201812

[41] Z. Xiang, C. Chu, and H. Chen. 2006. A fast heuristic for solving a large-scale static dial-a-ride problem under complex constraints. *European Journal of Operational Research* 174, 2 (2006), 1117–1139. https://doi.org/10.1016/j.ejor.2004.09.060