Multi-Model Ridesharing Problem

Hoon Oh[1], **Taoan Huang**[2], **Fei Fang**[1],
[1] Carnegie Mellon University
[2] Tsinghua University
hooh@andrew.cmu.edu, hta15@mails.tsinghua.edu.cn, feifang@cmu.edu

Abstract

The ridesharing problem has got increasing attention in recent years. Previous works studied dynamic ridesharing problem (requests arrive on-demand) and static ridesharing problem (request information are known in advance) separately. However, there is lack of work in combining dynamic and static ridesharing problem. In our work we study multi-model ridesharing problem (MMR), where we consider both pre-scheduled requests (stage 1) and on-demand requests (stage 2). We present (i) new complexity results for several MMR problems including the general and several special setting; (ii) an exact (MILP) and several heuristic approaches (Seq-MILP, Seq-DP) for stage 1 and an algorithm for stage 2 using matching algorithm; (iii) an extension of our algorithm for case with distribution of ride requests; (iv) extensive experimental results comparing runtime and solution quality for our solution approaches.

1 Introduction

Transportation is a significant component of the modern economy. However, modern transportation system still produces significant amount of pollution, congestion, and energy consumption. The ridesharing platform tries to resolve these problems by matching request demands and driver with idle cars. The platform matches these riders and drivers to reduce the cost. Recently the large-scale adoption of smart phones and significant decrease in cellular communication led to significant growth in the ridesharing market. Companies such as Uber and Lyft provide ridesharing platforms in many cities. However, Uber and Lyft consider very limited model of the problem. Users cannot specify their willingness to wait. Furthermore, it only considers on-demand requests to assign matching for drivers. Unlike urban areas, many suburban areas lack supply of drivers. However, there are still many ridesharing needs in suburban area. Therefore, riders may not able to guarantee to get a ridesharing service. Our multi-model ridesharing model will provide service for both pre-scheduled requests and on-demand requests to resolve these limitation.

In many suburban areas or school districts, there are many people who want to frequently travel between certain locations with a fixed schedule, i.e. students going to after-school program every day, going to grocery shopping every week. In these scenarios riders may want to pre-schedule for future ride and guarantee their rides within specific time window. Furthermore, our model combines pre-scheduled requests with on-demand requests where we want to have the ridesharing platform that satisfies all committed pre-scheduled requests, can also satisfy on-demand requests.

Recent ridesharing literature identifies dynamic ridesharing problems and static ridesharing problems. In dynamic ridesharing platforms, they study the problem where all the riders arrive on-demand. Whereas in static ridesharing problems, all riders' information are known in advance. Many previous works in dynamic ridesharing problems and static ridesharing problems are summarized in [Agatz *et al.*, 2012]. Several dynamic ridesharing models have been applied in real-world recently. However, static ridesharing algorithms have not been applied in real-world scenario as much as dynamic ridesharing problems.

Multi-model ridesharing combines both dynamic ridesharing problem and static ridesharing problem. Multi-model ridesharing problem can be seen as two-stage problem. Where in first stage, we get set of requests for the future, call them *pre-scheduled* requests. Then the algorithm commits to which requests to satisfy. Then in second stage, we get on-demand requests on top of pre-scheduled requests we computed in first stage. Then we want to satisfy on-demand requests subject to satisfying committed pre-scheduled requests. We compare and combine three different variants of the problems. We consider single-car vs multi-car case, single-capacity vs multi-capacity case, and impatient requests vs patient requests case. We study MMR problem with objective of maximizing value of satisfied requests. This objective can be interpreted as maximizing social welfare.

We show complexity results of several different cases of MMR problem. We show NP-hardness results for MMR problem for patient requests. We also show NP-hardness results for MMR problem for impatient requests on multi-car multi-capacity case. Although there are many studies in ridesharing problem, there was no NP-hardness result for impatient request case. Further our NP-hardness result for patient rider case is simpler model than previous NP-hardness

results.

We also give new algorithmic approaches for first-stage problem. We mainly focused on multi-car multi-capacity impatient model. However, the technique can be applied to any other model. We first formulated Mixed Integer Linear Program (MILP) using multi-commodity flow. Even though MILP will give exact solution, the running time grows exponentially as the number of drivers increases. Then we formulated dynamic programming (DP) for single-car multi-capacity impatient model. Then we solved the problem by running DP sequentially (Seq-DP), where each iteration greedily maximize the objective value. We also used same sequential approach using single-driver MILP (Seq-MILP). We compared the running time and solution quality of Exact-MILP, Seq-MILP and Seq-DP using simulation. We also show sequential approach gives $\frac{1}{2}$-approximation guarantee. The experiments shows the approach has experimental approximation ratio of 0.83. Furthermore, the running time of the seq-DP is significantly faster than exact-MILP approach.

Then we design an algorithm for second-stage problem. After choosing which requests to satisfy from first-stage. We solve the dynamic ridesharing problem subject to satisfying all chosen pre-scheduled requests. For dynamic ridesharing problem, we use algorithm similar to the one used in [Alonso-Mora *et al.*, 2017a]. They compute trips using sharability-graph, and run matching MILP to match drivers and sybset of on-demand requests. We modified the algorithm to capture pre-scheduled requests. Our improvement reduce number of possible set of requests, thus perform better than previous work for MMR problem. We ran experiments by simulation to see runtime and solution quality for our approach.

We also present data-driven optimization method. We formulated integer linear programming for data-driven case. In data-driven approach we choose pre-scheduled requests based on previous data, such that we take pre-scheduled trip that has less conflict with promising on-demand requests. The data-driven approach does not myopically optimize among pre-scheduled requests, thus it can satisfy more requests both stages combined. We also ran experiments by simulation to compare data-driven MILP's solution quality.

2 Related Work

Ridesharing has been increasingly important to enhance urban transportation sustainability and recently emerged as an active research topic. There are several pieces of work on dynamic ridesharing that are closely related to ours. In [Santos and Xavier, 2013], several heuristics are developed for slightly different objective and constraints. In [Alonso-Mora *et al.*, 2017a], a framework for real-time high-capacity ridesharing is presented and also incorporates rebalancing of idle cars to areas of high demand. The algorithm works well for setting with a short time frame but not for a long time horizon. But as we proposed, we can extend the algorithm to solve ridesharing with pre-scheduled requests. More previously work on dynamic ridesharing includes [Fabri and Recht, 2006] [Hasan *et al.*, 2018] and see [Agatz *et al.*, 2012] for an overview.

Another closely related problem is known as Dial a Ride Problem (DARP), [Cordeau and Laporte, 2007], where the problem is to transport people between locations with constraints such as time windows and ride time limits. The differences between DARP and ridesharing are riders do not share trips in DARP and mostly the services operate according to a static mode in practice. Some variants of DARP have been studied for different applications, where ridesharing with constraints is considered one of them. In [Santos and Xavier, 2015], [Santos and Xavier, 2015] and [Faye and Watel, 2016], they study ridesharing with money as an incentive. The main difference between ours and their work is that they imposed a client cost constraint. [Faye and Watel, 2016] studies static DARP with the goal to minimize the number of serving taxis among a given set of cars, where similar cost and time constriants are considered. We refer readers to [Cordeau and Laporte, 2007] and [Nedregård, 2015] for a review on DARP.

There are some pieces of work that include hardness results on ridesharing related problems. For ridesharing problems where riders share the cost, [Santos and Xavier, 2013] shows that the optimization problem to maximize the number of shared trips is NP-Hard. [Bei and Zhang, 2018] shows that to assign cars to requests with two requests sharing one car with minimum cost is NP-Hard and provides an 2.5-approximation algorithm. [Santos and Xavier, 2015] and [Faye and Watel, 2016] show that different cases of DARP with cost and constraints are NP-Hard. Although there are many works in complexity, NP-hardness of impatient requests case was not known. We show NP-Hardness for impatient ridesharing problem. Further we show NP-hardness for patient request for simple setting than previously studeid problem.

There are previous work that studied different types of multi-model ridesharing platform. [Mahéo *et al.*, 2017] proposes a bus network with hub and shuttle model. [Lam *et al.*, 2015] study crew routing problem where crews in the area are able to interchange cars. Their multi-model correspond to combination of different type of transportation or drivers. In our multi-model, we consider different models of requests.

We also look at data-driven optimization for the problem. There has been some studies of data-driven approach for ridehsharing problem. [Alonso-Mora *et al.*, 2017b] uses simple sampling methods to improve [Alonso-Mora *et al.*, 2017a] algorithm. Although their methods are simple and easy to implement, they only use single sampling for future prediction, thus it is vulnerable to high variance data. Our approach adopts the data-driven optimization to several scenarios of historical data to improve data-driven methods for the problem. [Guimarans *et al.*, 2015] run simluation and analyze data to predict request demand in port Botany. [Wen *et al.*, 2017] and [Lin *et al.*, 2018] use reinforcement technique to approach ridesharing related problem.

3 Model

We approach MMR as 2-stage problem. In the first stage, we get set of pre-scheduled requests . We commit to set of pre-scheduled requests to satisfy. Then in the second stage, we get on-demand requests. We want to maximize value of satisfied pre-scheduled requests and on-demand requests subject

# Cars	Capacity	Patient	Complexity	Exact Algorithm	Heuristic Algorithm
> 1	1	0	P [Ma *et al.*, 2018]	MCNF [Ma *et al.*, 2018]	
> 1	> 1	0	**Strongly NPC** [3SAT]		
1	> 1	1	**Strongly NPC** [3PART]	**MILP**	**Seq-MILP**, **Seq-DP**, Matching [Alonso-Mora *et al.*, 2017a]
> 1	1	1			
> 1	> 1	1			

Figure 1: Complexity and solution approaches table for stage 1 (Bold face represent our contribution)

# Cars	Capacity	Complexity	Algorithm	Data-Driven Algorithm
1	> 1	Strongly NPC [3PART]	RTVP-Matching	Data-Driven MILP
> 1	1			
> 1	> 1			

Figure 2: Complexity and solution approaches table for stage 2

to satisfying all committed pre-scheduled requests.

Let $G = (V, E)$ be the road network graph that represents the regions we consider in the ridesharing problem. Let $T, \mathcal{D}, \mathcal{R}$ be the length of time horizon, the set of drivers and the set of requests. We assume T is an integer and the time horizon is discritized, i.e. the set of time step is $\{0, 1, \ldots, T\}$.

Let $\mathcal{R} = \mathcal{R}_{OD} \cup \mathcal{P}$, where $\mathcal{R}_{OD}$ is the set of on-demand requests, and $\mathcal{P}$ is the set of pre-scheduled requests. Each request $r \in \mathcal{R}$ contains an origin $o_r \in V$, a destination $d_r \in V$, a value v_r, a request time t_r and a time window $W_r = [t_r, t_r + dist(r) + \Delta_r]$, where $dist(r)$ is the length of shortest path from o_r to d_r measured by time, and Δ_r is the maximum time the request is willing to wait. We assume $dist(r)$ is integer valued. The request arrive to region o_r at time t_r and want to go to d_r within W_r. If not mentioned, we assume all requests are willing to wait for Δ time, that is $\Delta_r = \Delta$ for all $r \in \mathcal{R}$. However, most of our works generalize to any window size. Let $t^\star$ be the time length that we get on-demand requests, i.e. at time t, we get on-demand requests that arrive at time $[t, t + t^\star]$. For each time t, let $[t, t + t^\star]$ be *on-demand window*. We will partition $\mathcal{P}$ into $\mathcal{P}_{cur}$ and $\mathcal{P}_{fut}$. Let $\mathcal{P}_{cur}$ be set of pre-scheduled requests that arrive within the on-demand window. Let $\mathcal{P}_{fut}$ be pre-scheduled requests that arrive after the on-demand window. Let *current requests* be $\mathcal{R}_{OD} \cup \mathcal{P}_{cur}$, i.e. set of on-demand requests and pre-scheduled requests in on-demand window. For each driver $d \in \mathcal{D}$, we have origin o_d, time t_d and capacity k_d. For notation we let $[N] := \{1, ..., N\}$ for any integer N.

In this paper, we investigate different cases of the problem and show their complexity and algorithmic approaches for the problem. Different models are summarized in the figure 1 and figure 2. We look at single-driver case vs multi-driver case, single-capacity case vs multi-capacity case (i.e. $k_d = 1$ or $k_d \leq k$), as well as impatient case vs patient case (i.e. $\Delta = 0$ or $\Delta \geq 0$).

4 Complexity Results

In this section, we will show the complexity of MMR. We will show NP-hardness for the stage 1 problem for patient request case and multi-car multi-capacity impatient case. Since stage 1 problem is special case of general MMR problem, the NP-hardness holds for general MMR problem.

4.1 Complexity Results for Impatient Requests

We show complexity results for solving MMR with impatient requests, i.e. $\Delta = 0$. For each case, we first show results for the first stage of the MMR problem, then show that those results also apply to the second stage problem.

For the multi-car single-capacity case ($D \geq 1$ and $k = 1$) in the first stage, [Ma *et al.*, 2018] showed a polynomial-time algorithm using max-cost network flow algorithm for stage 1. For stage 2 we can still construct max-cost network flow with lower bound on edges for committed pre-scheduled request. It is well known that max-cost network with lower bound can be solved in polynomial time. Thus MMR can be solved in polynomial time for multi-car single-capacity case.

However, it is strongly NP-Hard to solve the MMR problem for multi-car multi-capacity case ($D \geq 1$ and $k \geq 1$)

Theorem 1. *Solving the first stage of* MMR *problem with impatient requests is strongly NP-Hard for multi-car multi-capacity case.*

The proof is moved to appendix B

Corollary 1. *Solving the second stage of* MMR *problem with impatient requests is strongly NP-Hard for multi-car multi-capcity case.*

Proof. The NP-Hardness for solving the second stage follows from theorem 1.

□

4.2 Complexity Results for Patient Requests

We show that for solving MMR problem with patient requests ($\Delta \geq 0$) is strongly NP-Hard. We first show the strongly NP-Hardness for the first stage problem with single-car single-capacity ($k = 1$ and $D = 1$), then we show that the hardness result apply to the case with $k \geq 1$ and $D \geq 1$ in both stages.

Theorem 2. *Solving the first stage of* MMR *problem with patient requests is strongly NP-Hard even in single-car single-capacity case.*

Proof. We will show a reduction from 3PART problem, which is known to be a strongly NP-Hard problem. In an instance of 3PART, we are given $3n$ positive integers $a_1, \ldots, a_{3n}$ and a bound $B \in \mathbb{Z}^+$, such that $B/4 < a_i < B/2$ and $\sum_{i=1}^{3n} a_i = nB$. The problem is to determine whether the $3n$ numbers can be partitioned into n groups, each having exactly 3 numbers with the sum of each group equals B.

Given an instance of 3PART problem where B is polynomially bounded, we can construct a set $\mathcal{R}$ of $4n-1$ requests as follows. Consider the graph $G = (V, E)$ where $V = \{v_i : 0 \leq i \leq 4n-1\}$ and $E = \{(v_0, v_i) : i > 0\}$.

1. For request $1 \leq i \leq 4n-1$, it demands a trip from v_0 to v_i.
2. Let $[0, nB+n-1]$ be the time window of the i-th request $(1 \leq i \leq 3n)$.
3. Let $[i(B+1)-1, i(B+1)]$ be the time window of the $3n+i$-th request $(1 \leq i < n)$.

Let the length of edge (v_0, v_i) be $a_i/2$ for $1 \leq i \leq 3n$ and $1/2$ for $i > 3n$.

Since that $B \leq \text{poly}(n)$ and $\frac{B}{2} < a_i < \frac{B}{4}$, the bounds of all windows are polynomially bounded. Now we want to answer the question of whether there exists a schedule for the driver that satisfies $4n-1$ requests. Suppose there exists a 3-partition, then it is easy to verify that there is a feasible schedule for all requests. Conversely, if there is a feasible schedule for all requests, we are able to partition the numbers into n groups such that the sum of each group is exactly B. Since that $\frac{B}{2} < a_i < \frac{B}{4}$, we can infer that each group has exactly 3 numbers. As a result, solving this case is NP-Complete in the strong sense. □

Corollary 2. *Solving the* MMR *problem for patient requests is strongly NP-Hard in both stages.*

Proof. The NP-Hardness for solving the MMR problem for patient requests with $k \geq 1$ and $D \geq 1$ follows from theorem 2. □

5 Solution approach

In this section we provide exact and heuristic algorithms for MMR problem. To solve MMR, we first need to solve stage 1. In 5.1, we provide different solution approaches for stage 1. We first provide Mixed Integer Programming formulation (MILP) to solve the problem exactly. Then we solve the single-car version of MILP sequentially to get heuristic solution (Seq-MILP). Similarly, we also provide Dynamic Programming approach to get heuristic solution (Seq-DP). We show the sequential approach has an approximation guarantee of $1/2$. Then in 5.2, we present algorithm to solve stage 2. Lastly in 5.3, we propose a data-driven integer programming method to solve stage 1 problem using historical on-demand requests data.

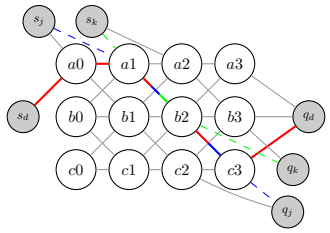


Figure 3: G_{TLG} of path graph with 3 discritized location a, b, c. The gray nodes represent virtual sources and sink nodes for the driver d, and the riders j and k. The red solid edges represent the driver's flow, the blue dashed line represent request flow of rider j, green dashed line represent request floe of rider k.

5.1 Algorithms for Solving Stage 1

Mixed Integer Programming

We will first show a mixed integer linear programming formulation for multi-car multi-capacity model. We will construct a time-location graph $G_{TLG} = (V, E)$ from input graph $G = (V_G, E_G)$. Where $V = \{(v, t) : v \in V_G, t \in [T]\}$, and we have edge between (v, t) and $(v, t+1)$ $\forall v \in V_G$ and $\forall t \in [T-1]$, and we have edge between (v, t) and $(u, t + dist(v, u))$ $\forall u, v \in V_G$ and $\forall t \in [T - dist(u, v)]$. We can interpret vertex (v, t) as driver (or rider) is at location v at time t.

We will have variable x for each driver flow. We will have variable y for each request flow. We will have variable z for satisfied request. More formally, $x_{ie} = 1$ if the car i goes through the edge e. $y_{jie} = 1$ if the request j goes through the edge e with the car i. $z_j = 1$ if the request j is satisfied. We also add virtual vertices s_j and q_j $\forall j \in \mathcal{R}$ which correspond to source and sink for each request flow. s_j has edge to (o_j, t) $\forall t \in [t_j, t_j + \Delta_j]$. Similarly, q_j has edge from (d_j, t) $\forall t \in [t_j + dist(j), t_j + dist(j) + \Delta_j]$. Let S_j be the set of s_j's neighbors and let Q_j be the set of q_j's neighbors.

Figure 3 shows example for TLG graph. Where G is a path graph with 3 nodes (a, b, c), and $T = 4$. We have a request j that wants to go from a to c at time 0. We also have a request k that wants to go from a to b at time 1, with $\Delta_j = \Delta_k = 1$. We have a driver starts from a at time 0 with capacity 2. Gray nodes correspond to virtual sources and sinks. Red edges correspond to driver flow, and blue and green edges correspond to request flow in an optimal solution.

The MILP formulation is defined in Figure 4. Solving this MILP gives an exact solution of the problem. However, solving MILP is NP-hard, and the running time of this MILP grows significantly as the number of driver increases. However, the integarity gap of the LP seems to be close to Δ, which is small in practice. Therefore, getting integer solution from fractional solution is relatively faster than other intuitive MILP formulations.

Sequential-MILP Approach

We observed the running time of the MILP increases significantly as the number of drivers increases. However, running

$$
\begin{aligned}
\max \quad & \sum_{j\in\mathcal{R}} v_j z_j \\
\text{s.t.} \quad & \sum_{e\in\delta^+(v)} x_{ie} = \sum_{e\in\delta^-(v)} x_{ie} && \forall i\in\mathcal{D}, \forall v\in V \\
& x_{i0} = 1, \quad \sum_{t\in E_T} x_{it} = 1 && \forall i\in\mathcal{D} \\
& \sum_{e\in\delta^+(v)} y_{jie} = \sum_{e\in\delta^-(v)} y_{jie} && \forall i\in\mathcal{D}, \forall j\in\mathcal{R}, \forall v\in V \\
& y_{ji(s_j,v)} \le \sum_{e\in\delta^+(v)} y_{jie} && \forall i\in\mathcal{D}, \forall j\in\mathcal{R}, \forall v\in S_j \\
& y_{ji(v,q_j)} \le \sum_{e\in\delta^-(v)} y_{jie} && \forall i\in\mathcal{D}, \forall j\in\mathcal{R}, \forall v\in Q_j \\
& y_{jie} \le x_{ie} && \forall i\in\mathcal{D}, \forall j\in\mathcal{R}, \forall e\in E \\
& \sum_{j\in\mathcal{R}} y_{jie} \le k_i x_{ie} && \forall i\in\mathcal{D}, \forall e\in E \\
& z_j \le \sum_{i\in\mathcal{D}} \sum_{v\in S_j} y_{ji(s_j,v)} \le 1 && \forall j\in\mathcal{R} \\
& z_j \le \sum_{i\in\mathcal{D}} \sum_{v\in Q_j} y_{j(v,q_j)} \le 1 && \forall j\in\mathcal{R} \\
& y_{jie} \in \{0,1\} && \forall i\in\mathcal{D}, \forall j\in\mathcal{R}, \forall e\in E \\
& z_j \in \{0,1\} && \forall j\in\mathcal{R} \\
& x_{ie} \in \{0,1\} && \forall i\in\mathcal{D}, \forall e\in E
\end{aligned}
$$

Figure 4: MILP formulation

MILP for single-car is relatively fast. We suggest new heuristic approach called *Seq-MILP*, where we sequentially solve single-car multi-capacity problem. More formally, given an instance with $|\mathcal{D}|$ drivers. Then we first solve the problem with just one car. Then remove all satisfied requests. We repeat this process for $|\mathcal{D}|-1$ more times for each car. In experiment, this approach shows high solution quality and much faster runtime.

Sequential-DP approach

We give an efficient polynomial time algorithm that provides suboptimal solution assuming that the capacity k is a constant. In this algorithm, the algorithm assigns request to cars sequentially as Seq-MILP does.

For the instance with one driver, let e_r be the possible earliest pickup time by the car. First sort the requests in increasing order of $e_r - \lambda v_r$, where $\lambda > 0$ is a regularization coefficient. Let $dp(j,t,l,S)$ be the maximum total value that can be achieved among the first j requests at time t and location l with the set of passengers S on the ride, where $|S| \le k$.

As shown in Algorithm 1, the transition from $dp(j,t,l,S)$ to other states could be computed by considering whether request j is picked up at time t location l, where the location of the driver at time $t+1$ is, and whether passengers in S are dropped off. After computing whole DP table, we trace back from the optimal final state to get the assignments and route plan for the driver. The number of state is $O(|\mathcal{R}|^{k+1}T|V|)$, with careful implementation, the overall time complexity is $O(|\mathcal{R}|^{k+1}T|E|)$.

With multiple cars, we assign requests to cars by running the algorithm 1 sequentially for each driver.

Algorithm 1 Seq-DP

for each driver $d \in \mathcal{D}$ **do**
  $dp(0, t_d, o_d, \emptyset) = 0$
  sort $\mathcal{R}$ in increasing order of $e_r - \lambda v_r$
  for $t = t_d$ to T, $j = 0$ to $|\mathcal{R}| - 1$ **do**
    for each available state $dp(j,t,l,S)$ **do**
      if request $j+1$ can be picked up at (t,l) **then**
        $dp(j+1,t,l,S\cup\{j\})$ ←-- $dp(j,t,l,S)$
        ▷ Where $dp(\cdot)$ ←-- v means, $dp(\cdot) \leftarrow v$ if $v > dp(\cdot)$
      for $(l,l') \in E_G$ **do**
        $X \leftarrow$ request should be dropped off at l'
        $dp(j, t + dist(l,l'), l', S \setminus X)$ ←-- $dp(j,t,l,S) + val(X)$
      $dp(j+1,t,l,S)$ ←-- $dp(j,t,l,S)$
      $dp(j,t+1,l,S)$ ←-- $dp(j,t,l,S)$
  $l = \arg\max_{l'\in V_G} dp(|\mathcal{R}|, T, l', \emptyset)$
  Trace back from $dp(|\mathcal{R}|, T, l, \emptyset)$ get the assignments $\mathcal{R}_d$ and route plan for drive d
  $\mathcal{R} \leftarrow \mathcal{R} \setminus \mathcal{R}_d$

Analysis of Sequential Approach

In Seq-MILP approach, we are computing single-car problem sequentially to solve multi-car problem. In this section we will provide an approximation guarantee of running optimal single-car algorithm sequentially for multi-car problem. Notice this is not a submodular maximization problem (max-coverage problem), because the set of requests that driver 1 can satisfy is different than the set of requests driver 2 can satisfy. Therefore, the problem is different from the classic submodular maximization problem.

Theorem 3. *Sequentially running optimal single-car algorithm greedily (Seq-MILP) leads to $\frac{1}{2}$-approximation for multi-car problem.*

Proof. Let OPT be a the set of requests that a fixed optimal solution satisfy. Let OPT_i be the set of request that i satisfy in the optimal solution. Let ALG's be defined similarly for algorithm solution. Let $v(S) := \sum_{r\in S} v(r)$. Let n be the number of drivers. Note no rider r belongs to OPT_i and OPT_j at same time for all $i \neq j$. Then we have $v(\mathsf{OPT}) = \sum_{i=1}^n v(\mathsf{OPT}_i)$ and $v(\mathsf{ALG}) = \sum_{i=1}^n v(\mathsf{ALG}_i)$.

It is easy to see that $v(\mathsf{OPT}_1) \le v(\mathsf{ALG}_1)$. Similarly observe $v(\mathsf{OPT}_2) \le v(\mathsf{ALG}_2) + v(\mathsf{OPT}_2 \cap \mathsf{ALG}_1)$. The second driver could choose the path that satisfy OPT_2 but satisfying ALG_2 gives more value than satisfying left over riders in OPT_2. Similarly we can get $v(\mathsf{OPT}_i) \le v(\mathsf{ALG}_i) + \sum_{j<i} v(\mathsf{OPT}_i \cap \mathsf{ALG}_j)$. Now we will sum up the inequalities over all drivers. Then we get the following.

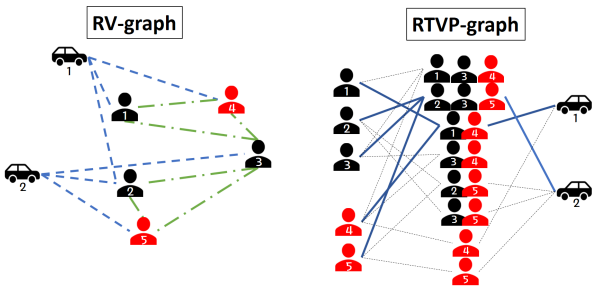


Figure 5: An example of RV-graph and RTVP-graph. Black cars correspond to driver nodes, black humans correspond to current requests, red humans correspond to future pre-scheduled request nodes. Solid lines in RTVP-graph correspond to an optimal matching.

$$\begin{aligned}\sum_{i=1}^{n} v(\mathsf{OPT}_i) &\leq \sum_{i=1}^{n} v(\mathsf{ALG}_i) + \sum_{i=1}^{n}\sum_{j<i} v(\mathsf{OPT}_i \cap \mathsf{ALG}_j) \\ &\leq v(\mathsf{ALG}) + \sum_{j=1}^{n-1}\sum_{i=1}^{n} v(\mathsf{OPT}_i \cap \mathsf{ALG}_j) \\ &\leq v(\mathsf{ALG}) + \sum_{j=1}^{n-1} v(\mathsf{ALG}_j) \leq 2v(\mathsf{ALG})\end{aligned}$$

Since a rider cannot be in OPT_i and OPT_j simultaneously, $\sum_{i=1}^{n} v(\mathsf{OPT}_i \cap \mathsf{ALG}_j)$ is at most $v(\mathsf{ALG}_j)$. Note left hand side is equal to $v(\mathsf{OPT})$. Therefore we get $\frac{1}{2}v(\mathsf{OPT}) \leq v(\mathsf{ALG})$, thus the algorithm always gives the solution that is at least half of an optimal solution, thus proving the algorithm is $\frac{1}{2}$-approximation.

□

5.2 Algorithms for Solving Stage 2

Now we will discuss heuristic algorithms to solve second-stage problem. In this stage, we are given on-demand requests within the on-demand window, and pre-scheduled requests $\mathcal{P}_{cur}$ and $\mathcal{P}_{fut}$. Our objective is to maximize total value of satisfied requests while satisfying all committed pre-scheduled requests and ensuring each future request is matched to exactly one driver.

We will closely follow Trip-Vehicle Matching algorithm similar to [Alonso-Mora *et al.*, 2017a]. In their work, they first construct RV-graph. In the RV-graph, we have vertices for riders and drivers. We have edges between a driver and a rider if the driver can satisfy the rider; we have edges between two riders, if a virtual driver starting from one of their origin can satisfy both riders. Then they construct RTV-graph by looking at a clique that contains a driver and construct Trip vertex. Then they run matching integer linear programming between drivers and feasible trips.

For our approach, we first construct RV-graph similar way. We treat pre-scheduled request the same way as on-demand requests in this process. Then we construct *RTVP-graph*. We have a vertex for each pre-scheduled request. We look for clique that contains at least one car and at least one future pre-scheduled request, and check whether that is feasible or not. This approach reduces the number of cliques we need to consider for the algorithm. A clique without future requests implies driver satisfying this trip may not be able to satisfy any pre-scheduled request, thus result in infeasible solution. If we have a driver without committed pre-scheduled route, then we will have an $\emptyset$ node as a pre-scheduled node. This correspond to one driver does not have to be matched to pre-scheduled request. Figure 5 shows an example of RV-graph and RTVP-graph for 2 cars, 3 current requests, and 2 future pre-scheduled requests.

After constructing RTVP-graph, we will solve matching integer linear programming. Let $\epsilon_{ijp} = 1$ if car i takes trip j that can reach future pre-scheduled request p. Then we have constraints where all future pre-scheduled requests are reachable and all current pre-scheduled requests are satisfied. In other words we will have following constraints. Let $\mathcal{T}$ be set of trips that our graph output.

$$\begin{aligned}
\max \quad & \sum_{i\in\mathcal{D}}\sum_{j\in\mathcal{T}}\sum_{p\in\mathcal{P}} v_j \epsilon_{ijp} & \\
\text{s.t.} \quad & \sum_{j\in\mathcal{T}}\sum_{p\in\mathcal{P}} \epsilon_{ijp} \leq 1 & \forall i \in \mathcal{D} \\
& \sum_{i\in\mathcal{D}}\sum_{j:r\in j} \epsilon_{ijp} \leq 1 & \forall r \in \mathcal{R} \\
& \sum_{i\in\mathcal{D}}\sum_{p\in\mathcal{P}_{fut}}\sum_{j:p'\in j} \epsilon_{ijp} = 1 & \forall p' \in \mathcal{P}_{cur} \\
& \sum_{i\in\mathcal{D}}\sum_{p\in\mathcal{P}_{fut}} \epsilon_{ijp} = 1 & \forall p \in \mathcal{P}_{fut} \\
& \epsilon_{ijp} \in \{0,1\} & \forall i \in \mathcal{D}, \forall j \in \mathcal{T}, \forall p \in \mathcal{P}_{fut}
\end{aligned}$$

First constraint ensure a car takes at most one trip. Second constraint ensure each request is satisfied at most once. Third and fourth constraints ensure current pre-scheduled request and future pre-scheduled request is satisfied. This approach add more variables to the integer programming than RTV-matching; however, constructing RTVP graph is much faster due to implicit pruning on number of cliques. We compare RTV-matching algorithm with regularization and RTVP-matching algorithm by simulation on section 6.

5.3 Data Driven Optimization

In this section we look at data-driven optimization problem. Where in the first stage of multi-model ridesharing, we want to decide pre-scheduled requests that does not interfere high-demanded or high-valued time zone. In this approach we sample scenarios from historical distribution, and compute commitment to pick-up pre-scheduled requests using historical on-demand requests and known pre-scheduled requests.

In this setting, a set of pre-scheduled requests $\mathcal{R}_0$ are given. Assume that on-demand requests appear according to a distribution F and we consider H scenarios of on-demand requests. To begin with our approach using MILP with on-demand prediction, a set of on-demand requests $\mathcal{R}_h$ with size R are sample for scenario $h \in [H]$, where requests are drawn

Algorithm 2 Construction of RTVP-graph

$\mathcal{T} = \emptyset$
for each driver $d \in \mathcal{D}$ **do**
  for each future pre-scheduled request $p \in \mathcal{P}_{fut}$ **do**
    $\mathcal{T}_i = \emptyset \; \forall i \in \{1, ..., |\mathcal{R}|\}$
    [Add trips of size one]
    for $e = (r, d) \in E_{RV}$ and $e' = (r, p) \in E_{RV}$ **do**
      $\mathcal{T}_1 \leftarrow T = \{r\}$.
      $E \leftarrow E \cup \{(r, T), (T, d), (T, p)\}$.
    [Add trips of size two]
    for all $\{r_1\}$,$\{r_2\} \in \mathcal{T}_1$ and $(r_1, r_2) \in E_{RV}$ **do**
      if $(d, p, \{r_1, r_2\})$ is feasible **then**
        $\mathcal{T}_2 \leftarrow T = \{r_1, r_2\}$.
        $E \leftarrow E \cup \{(r, T), (T, d), (T, p)\}$.
    [Add trips of size i]
    for $i \in \{3, ..., |\mathcal{R}|\}$ **do**
      for all $T_1, T_2 \in \mathcal{T}_{i-1}$ with $|T_1 \cup T_2| = i$ **do**
        if $(d, p, T_1 \cup T_2)$ is feasible **then**
          $\mathcal{T}_k \leftarrow T = T_1 \cup T_2$
          $E \leftarrow E \cup \{(r_i, T) \forall r_i \in T\}$.
          $E \leftarrow E \cup \{(T, d), (T, p)\}$.
    $\mathcal{T} \leftarrow \cup_{i \in [|\mathcal{R}|]} \mathcal{T}_i$
return $G = (\mathcal{R} \cup \mathcal{P} \cup \mathcal{D}, E)$.

i.i.d. from F. In the h-th senario, the set of all requests are $\mathcal{R}_h \cup \mathcal{R}_0$.

We extend our MILP formulation to capture different scenario with different probabilities. Our objective function will correspond to maximizing expected social welfare in both stage 1 and sampled stage 2. Our Data-Driven MILP (DMILP) formulation is moved to appendix C. Similar to solving stage 1, sequentially running the Data-Driven MILP (Seq-DMILP) for each driver will lead to a suboptimal solution with $1/2$ approximation guarantee.

Alternative algorithm to solve the data driven case is sampling one scenario according to the distribution, and solve the optimization problem with the sampled scenario. Similar method was used in [Alonso-Mora *et al.*, 2017b]. In appendix D, we show an example that DMILP can perform two times better than Single-sampling method.

6 Experiments

We run simulation for 3 settings in a 4 core 2.5GHz PC with 8GB memory.

The first setting is the first stage of MMR problem with multi-capacity cars and patient requests over a long time horizon. The runtime and solution quality are measured for single-car and multi-car cases and compared among the Exact-MILP algorithm and heuristic algorithms including Seq-MILP, Seq-DP and RTV-matching [Alonso-Mora *et al.*, 2017a]. We also show the scalibilty of the Seq-DP approach.

The second setting is the multi-car multi-capacity case of the second stage of MMR problem with on-demand requests within the on-demand window. In this setting, a set of pre-scheduled trips $\mathcal{P}$ is given and we assume all the drivers are pre-assigned to a current-pre-scheduled trip and a future-pre-scheduled. Our goal is to maximize the total value of satisfied requests subject to all the pre-scheduled requests are assigned to a drivers (possibly reassigned to new drivers). We compare the runtime and solution quality between RTV-matching with regularization term and RTVP-matching.

In the third setting, we show the runtime and solution quality of the data-driven approach for solving the first stage MMR problem.

In all setting, we run experiment on a $L \times L$ grid graph with 1 unit of weight on each edge (i.e. taking 1 unit of time to cross the edge). There are D drivers in the system and for simplicity, all cars have same capacity k. If not mentioned, in all experiments both the origin and destination of requests are drawn uniformly, the request time is drawn uniformly over time horizon of length T, the weights of requests are proportional to the shortest travel distance (time).

6.1 Simulation Results for Stage 1

In this part, we consider the first stage problem where $k \geq 1$ and $\Delta \geq 0$. For the first experiment, we run a small scale of the problems with $L = 7, k = 4, T = 50$ for 1-car and 4-car case to compare the runtime and solution quality among different approaches including exact MILP, Seq-MILP, Seq-DP and RTV-matching. The number of requests R ranges from 10 to 50. For each $R = i(10 \leq i \leq 50)$, we run 10 test cases. In the 4-car case, the optimal solutions are computed for $R \leq 30$. The limit of number of cliques for each driver to explore in RTV matching is set to 6000. In Seq-DP, λ is set to 1.

As shown in figure 6(a)(b) and appendix E, our approaches (Seq-MILP and Seq-DP) outperform RTV matching significantly when $R > 30$, while the gaps of objective value between the optimal and Seq-MILP, Seq-DP and Seq-MILP are relatively small. For runtime comparison, RTV matching takes long time than Seq-MILP when $R > 20$ and Seq-DP is significantly faster than other algorithms. Also, for all the test cases that the optimal solution is computed, the approximation ratios are at least 0.83, which is much better than the $1/2$ guarantee.

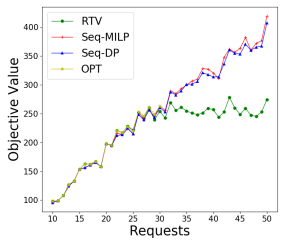
Then, to test the scalibilty of Seq-DP algorithm, we run eperiment having $D = 10$ drivers with capacity $k = 4$ each car on a $L \times L$ grid graph with 1 unit of weight on each edge. Set $T = 100$ for the time horizon and $R = 100$ for the number of requests.

We set a memory limit of 5GB and test L increasing from 5 until memory limit exceeded. For each $L = i$ $(5 \leq i \leq 22)$, we run 10 test cases and plot the runtime in figure 6(c). Within the memory limit, L can be as large as 22 and the average runtime for $L = 22$ is less than 1 minute.

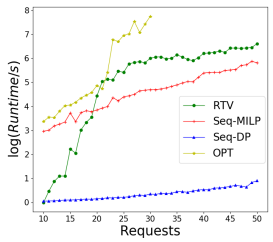
6.2 Simulation Results for Stage 2

We run experiments on a 7×7 grid graph and set $T = 50$. We have $D = 4$ drivers with capacity $k = 4$ for each car. Let $[0, t^*]$ be the on-demand window where $t^* = 16$.

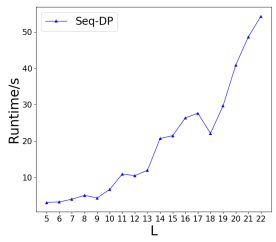
For pre-scheduled trips, we let $|\mathcal{P}_{cur}| = |\mathcal{P}_{fut}| = D$ and they are drawn the same way as on-demands (the time window for $\mathcal{P}_{fut}$ is $(t^*, T]$) and we ensure that there exists an one-to-one matching between drivers and future requests and



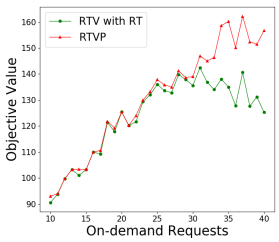
(a) Solution Quality for Multi-Car (L=7, D=k=4, T=50)



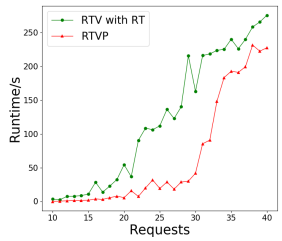
(b) Runtime for Multi-Car (L=7, D=k=4, T=50)



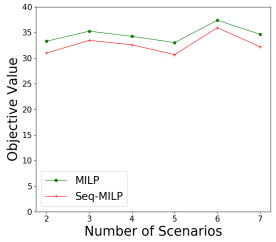
(c) Scalibilty for Seq-DP (D=10, k=4, T=100, R=100)



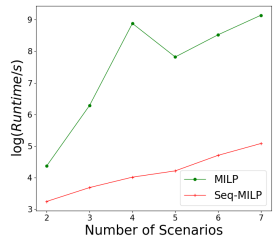
(d) Solution Quality for RTV and RTVP matching (L=7, D=k=4, T=50)



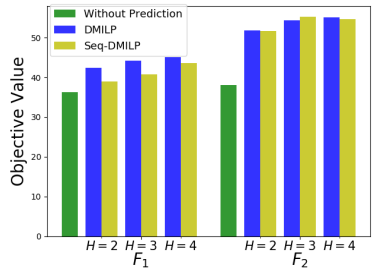
(e) Runtime for RTV and RTVP matching (L=7, D=k=4, T=50)



(f) Solution Quality for Demand Prediction (L=7, D=k=4, T=15)



(g) Runtime for Demand Prediction (L=7, D=k=4, T=15)



(h) Testing Result for Demand Prediction (L=7, D=k=4, T=15)

Figure 6: Experimental Results

there exist one feasible schedule that satisfy all pre-scheduled requests.

Let $S = \sum_{r \in \mathcal{R}_{QD}} v_r + 1$. By placing a regularization coefficient S on the value of pre-scheduled requests in the objective function, RTV matching is able to solve the exact same problem. We call this approach RTV matching with regularization term (RT). We compare the solution quality and runtime of this approach with those of RTVP matching.

For each $R = i$ $(10 \leq i \leq 40)$, we run 10 test cases. For each kind of clique in both algorithms, the limit of number of cliques to explore is set to 5000. As shown in figure 6(d)(e), the RTVP matching is faster than RTV matching with regularization term and outperforms RTV matching significantly for $R > 30$. When looking for cliques in RTVP matching algorithm, we are looking for cliques that contain a specific driver and a specific future request. The main reason why RTVP matching outperforms RTV matching is that we have stronger constraints for cliques in RTVP matching that bring down the number of feasible trips, while the RV graphs computed in both algorithms are the same.

6.3 Simulation Results for for Data-Driven Optimization

We run experiment on a 7×7 grid graph and set $T = 15$. We have $D = 4$ drivers with capacity $k = 4$ each car. We set $R = 10$ for each scenario.

We test the runtime and solution quality between DMILP and Seq-DMILP. For each $H = i$ $(2 \leq i \leq 7)$, we run 10 test cases. The results are shown in figure 6(f)(g) that the gap of objective value between two approaches is small while the sequential approach takes much shorter time for computation.

After computing committed pre-scheduled requests from (Seq-)DMILP, we test the solution quality by taking the average of sampling 5 other on-demand request sets. We test the algorithm with 2 distribution F_1, F_2 and $H = 2, 3, 4$. We lable the row and the column of the grid graph from 0 to 6 and label the vertex at row i column j as $7i + j$. For F_1, the origins of requests are drawn uniformly from row 0 to 2 and the destinations are drawn uniformly form row 4 to 6. For F_2, the origin of requests are drawn uniformly from vertices 0 to 9 and the destinations are drawn uniformly form vertices 39 to 48. For each distribution F_i and each H, we run 10 test cases. The results are shown in figure 6(h). For all distributions, as the number of scenario, H, increases, the solution quality with demand prediction gets better. Also the solution quality with demand prediction is higher than those without prediction.

7 Conclusion

In this paper, we studied MMR. We presented complexity results, solution approaches for both stage 1 and stage 2, and combining both stages using data-driven optimization. Our solution approaches scale to medium sized instances. Although this may be enough for suburban area, we want algorithmic approaches that scale to even larger instances. Also our experimental results are done by simulation. Running experiments with real-world data will help us verifying our algorithms solution quality and scalability for real-instances. We also assumed all time and location distance are integer valued. Extending the result to continuous valued problem

would reflect the real-world problem more. Also, in real-world we would like to compute pre-scheduled requests (for near future) and on-demand requests simultaneously. Although we can compute our algorithms within shorter window, there may be better approaches for online pre-schedule requests.

References

[Agatz *et al.*, 2011] Niels AH Agatz, Alan L Erera, Martin WP Savelsbergh, and Xing Wang. Dynamic ride-sharing: A simulation study in metro atlanta. *Transportation Research Part B: Methodological*, 45(9):1450–1464, 2011.

[Agatz *et al.*, 2012] Niels Agatz, Alan Erera, Martin Savelsbergh, and Xing Wang. Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research*, 223(2):295–303, 2012.

[Alonso-Mora *et al.*, 2017a] Javier Alonso-Mora, Samitha Samaranayake, Alex Wallar, Emilio Frazzoli, and Daniela Rus. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences*, 114(3):462–467, 2017.

[Alonso-Mora *et al.*, 2017b] Javier Alonso-Mora, Alex Wallar, and Daniela Rus. Predictive routing for autonomous mobility-on-demand systems with ride-sharing. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 3583–3590. IEEE, 2017.

[Bei and Zhang, 2018] Xiaohui Bei and Shengyu Zhang. Algorithms for trip-vehicle assignment in ride-sharing. 2018.

[Cordeau and Laporte, 2007] Jean-François Cordeau and Gilbert Laporte. The dial-a-ride problem: models and algorithms. *Annals of operations research*, 153(1):29–46, 2007.

[Fabri and Recht, 2006] Anke Fabri and Peter Recht. On dynamic pickup and delivery vehicle routing with several time windows and waiting times. *Transportation Research Part B: Methodological*, 40(4):335–350, 2006.

[Faye and Watel, 2016] Alain Faye and Dimitri Watel. Static dial-a-ride problem with money as an incentive: Study of the cost constraint. 2016.

[Guimarans *et al.*, 2015] Daniel Guimarans, Daniel Harabor, and Pascal Van Hentenryck. Simulation and analysis of container freight train operations at port botany. *CoRR*, abs/1512.03476, 2015.

[Hasan *et al.*, 2018] Mohd Hafiz Hasan, Pascal Van Hentenryck, Ceren Budak, Jiayu Chen, and Chhavi Chaudhry. Community-based trip sharing for urban commuting. 2018.

[Lam *et al.*, 2015] Edward Lam, Pascal Van Hentenryck, and Philip Kilby. Joint vehicle and crew routing and scheduling. In *International Conference on Principles and Practice of Constraint Programming*, pages 654–670. Springer, 2015.

[Lin *et al.*, 2018] Kaixiang Lin, Renyu Zhao, Zhe Xu, and Jiayu Zhou. Efficient large-scale fleet management via multi-agent deep reinforcement learning. *arXiv preprint arXiv:1802.06444*, 2018.

[Ma *et al.*, 2013] Shuo Ma, Yu Zheng, and Ouri Wolfson. T-share: A large-scale dynamic taxi ridesharing service. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 410–421. IEEE, 2013.

[Ma *et al.*, 2018] Hongyao Ma, Fei Fang, and David C. Parkes. Spatio-temporal pricing for ridesharing platforms. *CoRR*, abs/1801.04015, 2018.

[Mahéo *et al.*, 2017] Arthur Mahéo, Philip Kilby, and Pascal Van Hentenryck. Benders decomposition for the design of a hub and shuttle public transit system. *Transportation Science*, 2017.

[Nedregård, 2015] Ida Nedregård. The integrated dial-a-ride problem-balancing costs and convenience. Master's thesis, NTNU, 2015.

[Santos and Xavier, 2013] Douglas Oliveira Santos and Eduardo Candido Xavier. Dynamic taxi and ridesharing: A framework and heuristics for the optimization problem. In *IJCAI*, volume 13, pages 2885–2891, 2013.

[Santos and Xavier, 2015] Douglas O Santos and Eduardo C Xavier. Taxi and ride sharing: A dynamic dial-a-ride problem with money as an incentive. *Expert Systems with Applications*, 42(19):6728–6737, 2015.

[Wen *et al.*, 2017] Jian Wen, Jinhua Zhao, and Patrick Jaillet. Rebalancing shared mobility-on-demand systems: A reinforcement learning approach. In *Intelligent Transportation Systems (ITSC), 2017 IEEE 20th International Conference on*, pages 220–225. IEEE, 2017.

A Notation Table

In Figure 7 we summarized notations in our paper.

B Proof of Theorem 1

Proof. We will show reduction from 3SAT problem. In 3SAT we have n variables and m clauses. Each clause c_j ($j \in [m]$) contains 3 literals joint with logical or, for example $(x_i \vee \bar{x_j} \vee x_k)$. The formula contains logical and of all m clauses and we want to answer whether there exists a satisfying assignment for the formula. Now given 3-SAT formula, we will reduce it to the MMR problem.

Let $x_i(0) := \bar{x_i}$ and $x_i(1) := x_i$. $x_i(b)$ can be seen as assigning value of b to x_i. Let $c_k[j]$ be j-th literal in clause c_k, for example if $c_k = (x_2 \vee x_5 \vee x_7)$, then $c_k[1] = x_2$, $c_k[2] = x_5$, $c_k[3] = x_7$.

We will define vertices in the graph. For each variable x_i, we have vertices $i_b^k \quad \forall b \in \{0, 1\}$ and $k \in \{0, 1, ..., m\}$, and start vertex i_s. For each clause c_k, we have clause vertices $k_1, k_2, ..., k_6$. We will have n drivers corresponding to each variable. Each driver i starts at time -1 at vertex i_s. All drivers will have capacity 2. Let $dist(u, v)$ be the distance between u and v in the graph. Distance between two vertices

Notation	Meaning
$G = (V, E)$	Road Network
T	The length of the time horizon
$\mathcal{D}$	Set of drivers
$\mathcal{R}$	Set of requests
$\mathcal{R}_{OD}$	Set of on-demand requests
$\mathcal{P}$	Set of pre-scheduled requests
on-demand window	Time window that we receive on-demand requests
$\mathcal{P}_{cur}$	Set of pre-scheduled requests in on-demand window
$\mathcal{P}_{fut}$	Set of pre-scheduled requests after on-demand window
$o_r\ d_r\ v_r\ t_r\ \Delta_r$	Origin, destination, value, time, and waiting time for a request r
$dist(r)$	Length of shortest path from o_r to d_r measured in time
W_r	Window of the request $[t_r, t_r + dist(r) + \Delta_r]$
$o_d\ t_d\ k_d$	Origin, time, and capacity of a driver d
$t^\star$	The time length of on-demand window

Figure 7: Notation Table

is defined as follow for all $i \in [n]$, $b \in \{0, 1\}$, $k \in [m]$:

$$dist(i_b^{k-1}, i_b^k) = 7,$$
$$dist(i_b^{k-1}, k_j) = j \quad \text{if } x_i(b) = c_k[j],$$
$$dist(k_{j+3}, i_b^k) = 4 - j \quad \text{if } x_i(b) = c_k[j],$$
$$dist(k_j, k_{j+1}) = 1 \quad \forall j = \{1, ..., 5\},$$
$$dist(i_s, i_b^0) = 1.$$

Intuitively, the driver i starts from state i_s and if x_i is assigned to be 0, then the driver will take the path $(i_0^{k-1}, k_j, ..., k_{j+3}, i_0^k)$, for all c_k that x_i belongs. This will satisfy clause requests that will be defined later.

Now we will define requests. In high level, we have variable requests and clause requests. Variable requests ensure if the driver i chooses 0, then it stays at states i_0's. Clause requests ensure at least one literal is set to true in that clause. For each clause k if literal $x_i(b)$ is not in the clause, we have request that goes from i_b^{k-1} to i_b^k with value $7v$, where v is defined to be $M/7m$. If literal is in the clause at j-th index, then we have request that goes from i_b^{k-1} to k_j and k_j to i_b^k, with value jv and $(7-j)v$ respectively. Intuitively, if $x_i(b)$ is in the clause k, then from state i_b^{k-1} we want to go to k's clause vertices to satisfy requests. Lastly we have clause requests starting from k_3 to k_4 for all $k \in [m]$ with value 1. Note the values of variable requests are proportional to distance of the travel such that if we satisfy all requests corresponding to $x_i(b)$ we get total value of $(T-1)v = 7mv = M$.

The figure 8 shows an example with 3 variables (x_1, x_2, x_3) with clause $c_a = (x_1 \vee x_2 \vee x_3)$ and $M = 7m$. Each colored edge correspond to a request. Weight on edge correspond to both distance and value of the request, no weight correspond to unit distance (unit value). If one of 3 drivers choose path of 1, then it will go through edge (a_3, a_4) thus satisfying the clause request.

In this problem we want to answer the question whether there exists a schedule for drivers such that we satisfy total value of at least $nM + m$.

We will first show forward direction of the reduction. Given satisfying 3SAT assignment, we want to show there

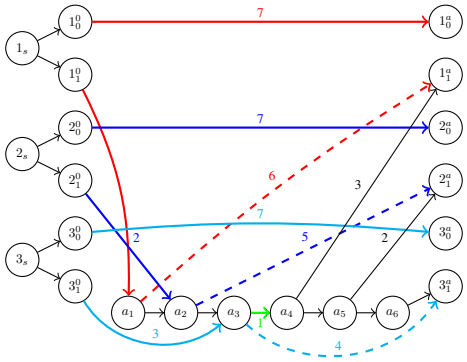


Figure 8: Graph at $c_a = (x_1, x_2, x_3)$

is schedule with a total value of $nM + m$. If $x_i = 0$ (or 1) in the satisfying assignment, then we will schedule driver i to follow path goes through all i_0^k's (i_1^k). Since following the path follows all the variable requests for that literal, it satisfy M values each. Furthermore, since it is a satisfying assignment, for all clause there is at least one driver goes through the clause request, thus the clause request is satisfied. Therefore, in total, the schedule satisfies a total value of $nM + m$.

Conversely, we show if there is a schedule of value at least $nM + m$, then there is a satisfying assignment for 3SAT instance. First note, there is no two variable requests that can be combined to satisfied together. Further, if any car is idle for some time (not satisfying any request), the driver cannot satisfy M variable requests. Lastly, there are m non-variable requests. Combining all these, in order to get total value of $nM + m$, each driver must satisfy in average of M variable requests. Since a driver can satisfy at most M variable requests, they all have to satisfy M variable requests, thus have to following intuitive path. Those path correspond to an assignment for each driver (thus an assignment for the variable). Therefore, any schedule for drivers with $nM + m$ value correspond to an assignment satisfying m clause requests, which leads to a satisfying assignment. □

$$\max \quad \sum_{h=0}^{H} p^h \cdot \left(\sum_{j \in \mathcal{R}_h} v_j z_j^h \right)$$

$$\text{s.t.} \quad \sum_{e \in \delta^+(v)} x_{ie}^h = \sum_{e \in \delta^-(v)} x_{ie}^h \qquad \forall i, v, h$$

$$x_{i0}^h = 1, \quad \sum_{t \in E_T} x_{it}^h = 1 \qquad \forall i, h$$

$$\sum_{e \in \delta^+(v)} y_{jie}^h = \sum_{e \in \delta^-(v)} y_{jie}^h \qquad \forall i, h, v, j \in \mathcal{R}_h^0$$

$$y_{ji(s_j,v)}^h \leq \sum_{e \in \delta^+(v)} y_{jie}^h \qquad \forall i, h, j \in \mathcal{R}_h^0, v \in S_j$$

$$y_{ji(v,q_j)}^h \leq \sum_{e \in \delta^-(v)} y_{jie}^h \qquad \forall i, h, j \in \mathcal{R}_h^0, v \in Q_j$$

$$y_{jie}^h \leq x_{ie}^h \qquad \forall i, h, e, j \in \mathcal{R}_h^0$$

$$\sum_{j \in \mathcal{R}_h \cup R_0} y_{jie}^h \leq k_i x_{ie}^h \qquad \forall i, h, e$$

$$z_j^h \leq \sum_{i \in \mathcal{D}} \sum_{v \in S_j} y_{ji(s_j,v)}^h \leq 1 \qquad \forall h, j \in \mathcal{R}_h$$

$$z_j^h \leq \sum_{i \in \mathcal{D}} \sum_{v \in Q_j} y_{ji(v,q_j)}^h \leq 1 \qquad \forall h, j \in \mathcal{R}_h$$

$$z_j^0 \leq \sum_{i \in \mathcal{D}} \sum_{v \in S_j} y_{ji(s_j,v)}^h \leq 1 \qquad \forall h, j \in \mathcal{R}_0$$

$$z_j^0 \leq \sum_{i \in \mathcal{D}} \sum_{v \in Q_j} y_{ji(v,q_j)}^h \leq 1 \qquad \forall h, j \in \mathcal{R}_0$$

$$y_{jie}^h \in \{0,1\} \qquad \forall i, h, e, j \in \mathcal{R}_h$$

$$z_j^h \in \{0,1\} \qquad \forall h \in [H]^0, j \in \mathcal{R}_h$$

$$x_{ie}^h \in \{0,1\} \qquad \forall i, e$$

Figure 9: Data-Driven MILP

C Data-Driven Mixed Integer Linear Programming (DMILP)

The formulation for data-driven MILP is shown in Figure 9. Let $x_{ie}^h = 1$ if the car i goes through the edge e. $y_{jie}^h = 1$ if request $j \in \mathcal{R}_h \cup \mathcal{R}_0$ at scenario h go through the edge e with car i. $z_j^h = 1$ if request $j \in \mathcal{R}_h$ is satisfied.

We also add virtual vertex s_j and q_j $\forall j \in \cup_{i=0}^{H} \mathcal{R}_i$ which correspond to source and sink for each request flow. s_j has edge to origin of j at time t_j to $t_j + \Delta_j$. Similarly, q_j has edge from destination of j at time $t_j + dist(j)$ to $t_j + dist(j) + \Delta_j$. Let S_j be neighbor of s_j and Q_j be neighbor of q_j. Let $\mathcal{R}_h^0$ be $\mathcal{R}_h \cup \mathcal{R}_0$. Similarly let $[H]^0$ be $[H] \cup \{0\}$. Unless otherwise stated, i is enumerated in $\mathcal{D}$, v in V, e in E, and h in $[H]$.

D Example of gap between DMILP and Single-Sampling

We will show DMILP can perform two times better than 1-sampling method. In the figure 10, we have 3 pre-scheduled requests of value $(1+\epsilon)$, 1, and $(1+\epsilon)$. We also have two sampled scenarios (red scenario and blue scenario). In any deterministic driver path, the driver can only satisfy one pre-scheduled requests. We will first analyze expected value single-sampling. In this algorithm, if we choose red (blue) scenario, then we will go to upper (lower) path u_1, u_2 (d_1, d_2), to get total expected value of $(1+\epsilon)+1/2v$, because red (blue) scenario happens with probability $1/2$. Whereas DMILP will choose center path c_1, c_2 considering both scenarios, and this gives expected value of $1 + 1/2v + 1/2v = 1 + v$. Therefore, DMILP perform two times better than 1-sampling algorithm in this example.

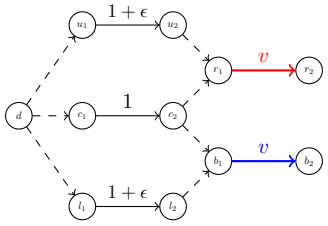
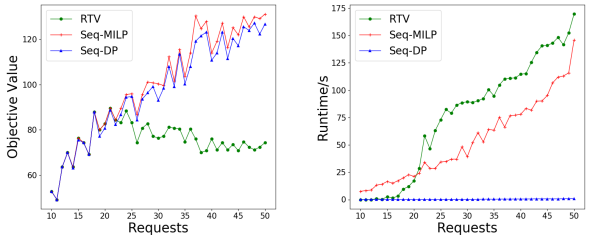


Figure 10: Multi-scenario sampling vs Single-sampling



(a) (b)

Figure 11: Single-car case(L=7, D=1, k=4, T=50)

E Experimental Result for Single-Car Case

In figure 11, (a) shows the solution quality and (b) shows the runtime comparison among Seq-MILP, Seq-DP and RTV matching, note that Seq-MILP gives the optimal solution in single-car case.