Appendices

A Proof of Lemma 1

Proof. By definition, c ends its service upon completing r. First we build up a transition graph $G' = (\mathcal{S}, E')$ among the set of possible states in the corresponding MDP, where $E' = \{(s, s') | \exists a \in \mathcal{A} \text{ s.t. } \mathcal{P}^a_{ss'} > 0\}$. Clearly, G' is a directed acyclic graph (DAG). Thus from Bellman Expectation Equation

$$v_\pi(s) = \sum_{a\in\mathcal{A}} \pi(a|s) \left(\mathcal{V}^a_s + \sum_{s'\in\mathcal{S}} \mathcal{P}^a_{ss'} v_\pi(s') \right),$$

we can see that there exists an optimal deterministic policy, $\pi(a^*|s) = 1$ where

$$\begin{aligned} a^* &= \arg\max_{a\in\mathcal{A}(s)} \mathcal{V}^a_s + \sum_{s'\in\mathcal{S}} \mathcal{P}^a_{ss'} v_\pi(s') \\ &= \arg\max_{a\in\mathcal{A}(s)} \mathcal{V}^a_s + \mathbb{E}_{\mathcal{R}_{t_a,l_a}} [v_\pi(t_a, l_a, \mathcal{R}_{t_a,l_a})], \end{aligned} \tag{2}$$

and the state value together with the optimal policy can be determined following the topological ordering of states in G'.

Next we show that $\mathsf{CST}(t, l|r) = \mathbb{E}_{\mathcal{R}_{t,l}} [v_\pi(t, l, \mathcal{R}_{t,l})]$ by induction on t. It holds for $t = t_r$ where $\mathsf{CST}(t_r, l_r|r) = 0$ as line 1 of Algorithm 1 shows. Assume it holds for all (t, l) with $t > t'$. Then we are to show it is correct for $t = t'$. By the induction hypothesis and Equation 2, the platform will always choose an available action a with the highest $\mathcal{V}^a_s + \mathsf{CST}(t_a, l_a|r)$. In line 5-7 and as visualized in Figure 2, the platform will look in the order of $a_1, \ldots, a_j$ and pick the first location a_i that is a destination of a request $r \in \mathcal{R}_{t',l}$. Otherwise, the driver will be guided to drive idly to d^*. Let $P_i = \Pr[X_{l,a_i,t'} \geq 1 | X_{l,a_k,t'} = 0, \forall k < i]$, thus we have

$$\begin{aligned} & \mathbb{E}_{\mathcal{R}_{t',l}} [v_\pi(t', l, \mathcal{R}_{t',l})] \\ = & \sum_{i=1}^{j} P_i \cdot (V_{l,a_i,t'} + \mathsf{CST}(t' + \delta(l, a_i), a_i|r)) \\ & + \Pr[X_{l,a_k,t'} = 0, \forall k \leq j] \cdot \mathsf{CST}(t' + \delta(l, d^*), d^*|r) \\ = & \mathsf{CST}(t', l|r). \end{aligned}$$

The last equality follows from line 8-13, which concludes the proof. It follows $q_{\pi^*}(s, a) = \mathcal{V}^a_s + \mathsf{CST}(t_a, l_a|r)$ as a corollary. □

B Algorithm for UPDATEPROBDIST

We show in Algorithm 5 the update of distribution. It calculates p_w's following the intuition as we introduced

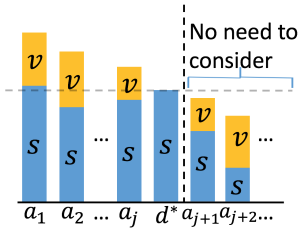

Figure 2: An Illustration of Algorithm 1. In the bar for a_i, s represents $\mathsf{CST}(t+\delta(l, a_i))$ and v represents $V_{l,a_i,t}$.

Algorithm 4 GETPROBABILITY$(l, t, x, h(\cdot)|r)$

```
1: if $l = l_r \wedge t = t_r$ then return
2: Retrieve the CST values $\mathsf{CST}(d, t+\delta(l, d)|r)$ for $d \in S(l, t|r)$
3: $d^* \leftarrow \arg\max_{d\in S(l,t|r)} \mathsf{CST}(d, t + \delta(l, d)|r)$
4: Denote $\{a_i\}$ the sequence of $d \in S(l, t|r)$ in decreasing order of $V_{l,d,t} + \mathsf{CST}(d, t + \delta(l, d)|r)$
5: $j \leftarrow$ the largest index of $\{a_i\}$ such that $V_{l,a_j,t} + \mathsf{CST}(a_j, t + \delta(l, a_j)|r) > \mathsf{CST}(d^*, t + \delta(l, d^*)|r)$
6: $p \leftarrow 1$
7: for $i = 1$ to $j$ do
8:     $w \leftarrow (l, a_i, t)$
9:     $p_w \leftarrow p_w + x \cdot p$
10:    GETPROBABILITY$(a_i, t+\delta(l, a_i), x\cdot p\cdot h(X_w \geq 1), h(\cdot)|r)$
11:    $p \leftarrow p \cdot (1 - h(X_w \geq 1))$
12: GETPROBABILITY$(d^*, t + \delta(l, d^*), x \cdot p, h(\cdot)|r)$
```

(line 1-3) and then the distribution is updated following Equation (1) (line 4-6). Algorithm 4 shows the calculation of p_w's, which follows a routine similar to calculating the CST value in Algorithm 1

C Proof of Theorem 2

Proof. Consider a graph of 4 vertices A, B, C, D. Let the weights of edges $\delta(B, D) = \delta(C, A) = \delta(A, D) = \mu$, $\delta(A, B) = \delta(D, C) = \mu - 1$ and $\delta(B, C) = t$ $(1 \leq t \leq \mu)$, which represent the number of time steps required to travel along the edges as shown in Figure 3. Suppose that $T = 4\mu - 2 + t$ and there is only 1 vehicle. The vehicle starts work at A at time 1.

Consider an instance on this graph and time horizon $[T]$ where we have 6 requests $r_1 = (B, C, 2\mu, t), r_2 = (A, D, 1, \mu), r_3 = (D, C, \mu + 1, \mu - 1), r_4 = (C, B, 2\mu, t), r_5 = (B, A, 2\mu + t, \mu - 1), r_6 = (A, D, 3\mu + t - 1, \mu)$.

The platform receives sufficiently many of r_1's at the beginning and the algorithm will take one of r_1's with a to-

Algorithm 5 UPDATEPROBDIST$(h(\cdot), a, r)$

```
1: Initialize $p_w \leftarrow 0$ for all $w \in \mathcal{W}$
2: $(l_a, t_a) \leftarrow$ the location-time pair action $a$ leads to
3: GETPROBABILITY$(l_a, t_a, 1, h(\cdot)|r)$
4: for $w \in \mathcal{W}$ do
5:     for $i = 1$ to $|\mathcal{D}|$ do
6:         $h(X_w \geq i) \leftarrow (1 - p_w) \cdot h(X_w \geq i) + p_w \cdot h(X_w \geq i + 1)$
7: return $h(\cdot)$
```

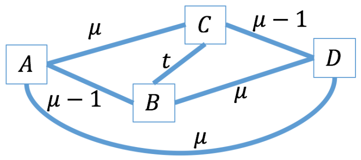

Figure 3: A Road Network Graph

tal revenue of t. This is because a deterministic algorithm should always accept the first feasible request, otherwise it would lead to a competitive ratio of ∞.

After that, $r_2, r_3, \ldots, r_6$ appear sequentially but the platform could accept none of them.

In this case, the offline optimal solution would have taken all the requests except r_1 to fill up the entire time horizon with a total revenue of $4\mu - 2 + t$. Thus we have the ratio to be at least $\frac{4\mu-2+t}{t}|_{t=1} = 4\mu - 1$.

□

D Proof of Theorem 3

Proof. Let R_{ALG} be the set of requests the First-Fit algorithm accepts and R_{OPT} be the one of the offline optimal solutions. For any request $r = (o_r, d_r, t_r, v_r) \in R_{\mathsf{ALG}}$, assume it is served by driver c. Let $t_{r,1} = t_r$ and $t_{r,2} = t_r + \delta(o_r, d_r)$. Consider a time interval $I = [t_{r,1} - (2\mu - 1), t_{r,2} + (2\mu - 1)]$.

We claim that the travel time interval of any request r' that is incompatible with request r and driver c, lies within I. This is because if the time interval of a request does not lie entirely in I, then the end time of this request should be no later than $t_{r,1} - \mu$ (start time of this request should be no earlier than $t_{r,2} + \mu$), thus it should be compatible with r.

Thus, the total value of the requests in R_{OPT} incompatible with r and driver c, is at most $4\mu - 2 + t_{r,2} - t_{r,1}$. Let $\mathsf{OPT} = \sum_{r \in \mathcal{R}_{\mathsf{OPT}}} v_r$ and $\mathsf{ALG} = \sum_{r \in \mathcal{R}_{\mathsf{ALG}}} v_r$, hence $\mathsf{OPT} \leq \sum_{r \in R_{\mathsf{ALG}}} (4\mu - 2 + t_{r,2} - t_{r_1})$. As a result,

$$\begin{aligned}
\frac{\mathsf{OPT}}{\mathsf{ALG}} &= \frac{\mathsf{OPT}}{\sum_{r \in R_{\mathsf{ALG}}} t_{r,2} - t_{r,1}} \\
&\leq \frac{\sum_{r \in R_{\mathsf{ALG}}} 4\mu - 2 + t_{r,2} - t_{r,1}}{\sum_{r \in R_{\mathsf{ALG}}} t_{r,2} - t_{r,1}} \\
&= \frac{|R_{\mathsf{ALG}}|(4\mu - 2)}{\sum_{r \in R_{\mathsf{ALG}}} t_{r,2} - t_{r,1}} + 1 \\
&\leq \frac{|R_{\mathsf{ALG}}|(4\mu - 2)}{|R_{\mathsf{ALG}}|} + 1 = 4\mu - 1.
\end{aligned}$$

□

E Offline Algorithm for Stage 1 without On-Demand Requests

The offline optimal solution of Stage 1 without on-demand requests can be obtained by solving a maximum cost network flow (MCNF).

Given the set of available vehicles $\mathcal{D}$ and all the scheduled requests $\mathcal{R}$, we construct a network $G = (V, E)$. We construct two vertices v_i and v'_i for each vehicle i, one entry-vertex $v_{r,in}$ and one exit-vertex $v_{r,out}$ for each request r, 2 virtual vertices S and T as the global source and sink.

We construct four types of edges in E. First, we construct edges from S to v_i and from v'_i to T, each with flow 1 and cost 0. Secondly, we construct edges from $v_{r,in}$ to $v_{r,out}$, with flow 1 and cost v_r, which mean each request could be taken no more than once. Thirdly, we construct edges from v_i to $v_{r,in}$ and from $v_{r,out}$ to v'_i, each with flow 1 and cost 0, which mean the first and last request the vehicle i could possibly served. Lastly, we construct edges from $v_{r_i,out}$ to $v_{r_j,in}$ if the distance between region i and region j allows a vehicle to pick up request j after serving request i.

Then by applying any MCNF algorithm, we could obtain the optimal solution.

F The Greedy-KM and Enhanced-KM

Greedy-KM works as follows. Given the set of available vehicles $\mathcal{D}$ and the state $(t_c, l_c, \mathcal{R}_{t_c,l_c})$ of each vehicle, we construct a bipartite graph $G_B = (\mathcal{D}, \bigcup_{c \in \mathcal{D}} \mathcal{R}_{t_c,l_c}, E_B)$, where we have edges between $c \in \mathcal{D}$ and $r \in \mathcal{R}_{t_c,l_c}$ with weight v_r. Greedy-KM dispatches order by finding a weighted maximum matching on G_B. In implementation, we employ the Kuhn-Munkres (KM) algorithm [Munkres, 1957] to solve it.

In Enhanced-KM, the bipartite graph is constructed in the same way as Greedy-KM, except that the edges between $c \in \mathcal{D}$ and $r \in \mathcal{R}_{t_c,l_c}$ have weight $v_r + \mathsf{CST}(t_c + \delta(o_r, d_r), d_r|\tilde{r_c})$, where $\tilde{r_c}$ is the next committed scheduled request of vehicle c.

G Learning & Planning Algorithm

The LPA is an adaptation of the work of Xu et al. [2018] to the hard constraints brought in by the scheduled requests.

In their work, they regard consider the transportation as the MDP and construct a local-view MDP for each driver, with location-time pairs as the states. As for the state transition rules and rewards for each state, they are drawn from the historical data. Actions of drivers are to pick up on-demand requests nearby or to stay still. For an action that lasts for T' time steps with reward R, they apply a discount factor γ and the final reward is given by

$$R_\gamma = \sum_{t=0}^{T'-1} \gamma^t \frac{R}{T'}.$$

At every time step, they obtain the value function v' for all states and then dispatch orders via a matching approach. The calculation of the value function is shown as Algorithm 6. We are using the same notation in Algorithm 6 as Xu et al. [2018] did, which do not have the same meaning as those in our main text.

We do the following to adapt their algorithms to our two-stage model. In Stage 1, we parse the scheduled requests and decide immediately for request r by the comparison of $R_\gamma(r) + V'(d_r, t_r + \delta(o_r, d_r))$ and $V'(o_r, t_r)$, meaning that a request would be accepted if it could lead to an increment in the expected value. In Stage 2, we will use the same reward function as the edge weights for all the possible state transitions. It is worth noting that we will forbid the driver to pick up an order whose ending time is too late for next scheduled request.

Moreover, Xu et al. [2018] mentioned the updates in Algorithm 6 can be done iteratively with the planning. The time window in our algorithm, however, is only one day. Thus this iteration cannot help optimize the value function.

H Clustered Centers in the City

In figure 4, we display the explicit real-world locations of 21 clustered centers in the Chinese city derived from the Didi data.

Algorithm 6 LPA

```
1: Collect all the historical state transitions $\mathcal{T} = \{(s_i, a_i, r_i, s'_i)\}$ from data; each state is composed of a time-location pair: $s_i = (t_i, location_i)$; each action is composed of the initial state and transited state: $a_i = (s_i, s'_i)$;
2: Initialize $V'(s)$, $N'(s)$ as zeros for all possible states.
3: for $t = T - 1$ to 0 do
4:     Get the subset $D^{(t)}$ where $t_i = t$ in $s_i$.
5:     for each sample $(s_i, a_i, r_i, s'_i) \in D^{(t)}$ do
6:         $N'(s_i) \leftarrow N'(s_i) + 1$
7:         $V'(s_i) \leftarrow V'(s_i) + \frac{1}{N'(s_i)}[\gamma^{\Delta t(a_i)} V'(s'_i) + R_\gamma(a_i) - V'(s_i)]$
8: Return the value function $V'(s)$ for all states.
```

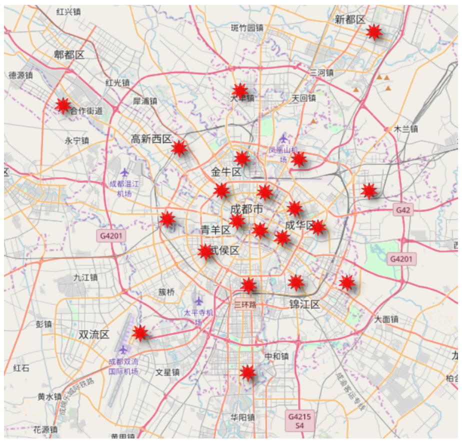

Figure 4: The Clustered Centers in a City in China derived from data.

I Single-Vehicle Case Performance

In Figure 5, we demonstrate the single-vehicle performances among all BESTSCORE, RANDOMBESTSCORE,BESTSCORE-A, BESTSCORE-R, RANDOMBESTSCORE-A, RANDOMBESTSCORE-R, and label them as BS, RBS, BS-A, BS-R, RBS-A, RBS-R respectively in the figure.

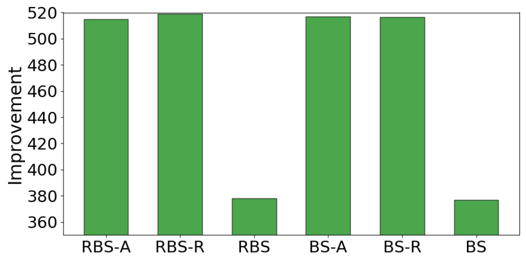

Figure 5: The Performance in Single-Vehicle Case.

J CST-Value as the Vehicle Number Increases

Here we empirically demonstrate how the CST-value changes as the number of vehicles increases. We fix a tuple of (t, l, r) and assume all vehicles start at (t, l) with r the next request to serve. Figure 6 shows how $\mathsf{CST}(t, l|r)$ changes and we can see that the decrease of $\mathsf{CST}(t, l|r)$ is quick at first and later slows down.

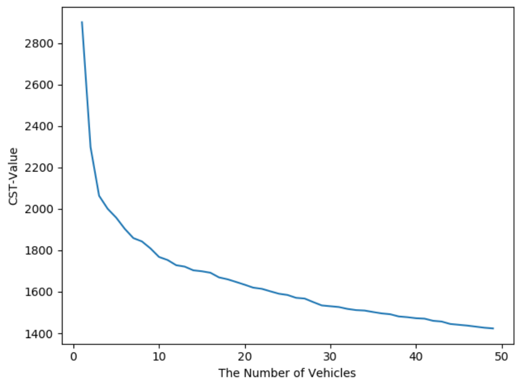

Figure 6: The Change of CST-function as Vehicles Increase.