Supplementary Material: A Deep Value-based Policy Search Approach for Real-world Vehicle Repositioning on Mobility-on-Demand Platforms

Yan Jiao¹, Xiaocheng Tang¹, Zhiwei (Tony) Qin¹, Shuaiji Li¹, Fan Zhang², Hongtu Zhu², Jieping Ye ²
¹DiDi Research America, Mountain View, CA 94943
²Didi Chuxing, Beijing, China
{yanjiao,xiaochengtang,qinzhiwei,shuaijili,feynmanzhangfan,zhuhongtu,yejieping}@didiglobal.com

1 The DPE Algorithm

The main procedure of DPE is presented in Algorithm [1]. We use the same discount factor $\gamma = 0.92$ in DPE (for computing the state values) and in VPS (for computing the path values). For the cerebellar embedding we use 3 quantization functions and a memory size $A$ of 20000. The embedding dimension $m$ is chosen to be 50. Following the cerebellar embedding layer are fully connected layers having $[32, 128, 32]$ hidden units with ReLU activations. To evaluate the policy we apply Adam optimizer with a constant step size $3e^{-4}$ and the Lipschitz regularization parameter $\lambda$ as $1e^{-4}$.

2 Dispatch Probability Model

We describe the experiment configuration for the results in Table [1]. For each city, we train a LightGBM decision tree, and use it to predict the probability of a driver receiving trip request given his or her current state. State features include driver_ids, driver’s location, time, and day of the week. We hash and encode each driver_id into a 5 dimensional dense vector, and treat each element of the vector as an input feature. Hyper-parameters are tuned via Bayesian optimization, where we fit a Gaussian process on target hyper-parameters, and use it as a surrogate model to optimize test AUC. We report the best hyper-parameter configuration in Table [2].

<table>
<thead>
<tr>
<th>City</th>
<th>Recall</th>
<th>Precision</th>
<th>F1</th>
<th>Accuracy</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.7782</td>
<td>0.7596</td>
<td>0.7835</td>
<td>0.8014</td>
<td>0.876</td>
</tr>
<tr>
<td>B</td>
<td>0.7568</td>
<td>0.7592</td>
<td>0.7618</td>
<td>0.7761</td>
<td>0.853</td>
</tr>
<tr>
<td>C</td>
<td>0.7745</td>
<td>0.7834</td>
<td>0.7812</td>
<td>0.7977</td>
<td>0.8729</td>
</tr>
</tbody>
</table>

Table 1: Evaluation results of the dispatch probability models.

<table>
<thead>
<tr>
<th>boosting_type</th>
<th>num_leaves</th>
<th>max_bin</th>
</tr>
</thead>
<tbody>
<tr>
<td>gbdt</td>
<td>6500</td>
<td>8500</td>
</tr>
<tr>
<td>min_data_in_leaf</td>
<td>26</td>
<td>0.1846772</td>
</tr>
<tr>
<td>lambda_l2</td>
<td>0.01846772</td>
<td>0.0834044</td>
</tr>
</tbody>
</table>

Table 2: Hyper-parameter configuration of the dispatch probability models.

*corresponding author

Algorithm 1 Dual Policy Evaluation (DPE) with Cerebellar Value Network (CVNet)

1: Given: historical driver trajectories \( \{ (s_i,0,o_i,0,r_{i,1},s_{i,1},o_{i,1},r_{i,2},...,r_{i,T_i},s_{i,T_i}) \} \in \mathcal{H} \) collected by executing a (unknown) policy \( \pi \) in the environment.
2: Given: \( n \) cerebellar quantization functions \( \{ q_1,...,q_n \} \), regularization parameter, max iterations, embedding memory size, embedding dimension, memory mapping function, discount factor \( \lambda, N, A, m, g(\cdot), \gamma \).
3: Compute training data from the driver trajectories as a set of (state, reward, next state) tuples, e.g., \( \{ (s_{i,t},R_{i,t},s_{i,t+k_i,t}) \in \mathcal{H}, t=0,...,T_i \) where \( k_i,t \) is the duration of the trip.
4: Initialize the embedding weights \( \theta_M \).
5: Initialize the state value network \( V(s) \) with random weights \( \theta_1 \).
6: Initialize the conditional state value network \( V(s|b) \) with weights \( \theta_2 \).
7: for \( \kappa = 1, 2, ..., N \) do
8: Sample a random mini-batch \( \{ (s_{i,t},R_{i,t},s_{i,t+k_i,t}) \} \) from the training data.
9: Embed the states \( s_{i,t} \) and \( s_{i,t+k_i,t} \) using the quantization functions \( \{ q_1,...,q_n \} \) and the memory mapping function \( g(\cdot) \), e.g., \( s \leftarrow c(s)^T \theta_M/n \) where the activation vector \( c(s) \) is initialized to 0 and iteratively adding 1 to the \( g(q_i(s)) \)-th entry of \( c(s) \) for \( i = 1,...,n \).
10: Set the binary option indicating idle movement or dispatch \( b \leftarrow 1_{R_{i,t} \geq 0} \).
11: Update the state value network \( V(s) \) with inputs \( s_{i,t} \) and targets \( R_{i,t} \gamma^{k_i,t} V(s_{i,t+k_i,t}) + \gamma^{k_i,t} V(s_{i,t+k_i,t}) \).
12: Update the conditional state value network \( V(s|b) \) with inputs \( [s_{i,t}; b] \) and targets \( R_{i,t} \gamma^{k_i,t} V(s_{i,t+k_i,t}) + \gamma^{k_i,t} V(s_{i,t+k_i,t}) \).
13: end for
14: return \( V \)

Figure 1

### 3 Expansion Depth Selection for VPS

We benchmark implementations of VPS with expansion depth = 1 (Greedy), 2, and 3 through simulation with data from a different city other than the three cities in the experiment section of the paper. Each method was tested using 10 random seeds, and we report both mean and standard deviation of the income rates. According to Figure 1, we have chosen an expansion depth of 2 for both the simulation and the real-world experiments based on average performance and robustness.

### 4 Real-world Experiment Setup

Since our experiments were run on the regular MoD platform with human drivers, the experiment required some careful set-up. We describe below the delivery of the repositioning tasks to the drivers and the incentive design for the program, which are crucial for the successful running of the
<table>
<thead>
<tr>
<th>City</th>
<th>algo Group</th>
<th>human Group</th>
<th>Population Size (millions)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>9</td>
<td>8</td>
<td>4.51</td>
</tr>
<tr>
<td>B</td>
<td>14</td>
<td>13</td>
<td>7.31</td>
</tr>
<tr>
<td>C</td>
<td>9</td>
<td>8</td>
<td>4.65</td>
</tr>
</tbody>
</table>

Table 3: The distribution of the experiment groups across the three cities and their population sizes.

experiment. The distribution of the drivers across the three cities and the cities’ population sizes are summarized in Table 3.

4.1 Repositioning Task Delivery

Repositioning recommendations are delivered through pop-up message cards within the mobile driver app. Once repositioning is triggered, a message card appears at the target driver’s app. The message card contains instructions for the repositioning task, including the destination and the target time that the driver is required to be there. After the driver acknowledges the task, GPS navigation is launched to provide turn-by-turn route guidance to the driver. The system automatically determines if the driver has reached the prescribed destination within the required time frame.

4.2 Incentive Design

Since our experiment’s goal is to benchmark algorithms on long-term cumulative metrics (daily income rate) and the supply-demand context could vary significantly within a day, it would be ideal if all the drivers in the program are online for the same period time which is also sufficiently long, and the drivers always follow the repositioning recommendations, for a fair comparison. We designed an incentive scheme to encourage the drivers to participate as closely to the ideal situation as possible. Specifically, we required that they are online for at least five hours out of the eight hours from 11am to 7pm (the experiment interval) during weekdays and that they skip or fail to complete no more than three tasks each day. The drivers were rewarded for each repositioning task that they finished, and they received additional reward for each day that they met the daily requirements and the week that they met daily requirements for all weekdays. Income rates are computed using the data where daily requirements are met.