Dynamic Pricing and Dispatching in Ride-Sharing Networks: A Transfer Reinforcement Learning Approach

T. De Munck, P. Chevalier and J.-S. Tancrez

Center for Operations Research and Econometrics, UCLouvain

August 31st, 2023

Outline

1. Motivations.
2. Markov decision process (MDP) with pricing and dispatching decisions.
3. Solution approaches:
   - Mathematical optimization.
   - Deep reinforcement learning (DRL).
4. Numerical study based on NYC dataset.

1. Motivation

Ride-sharing platforms are becoming critical components of urban transit.

- Uber: 7.6 billion trips in 2022, 131M monthly active customers, 3.5M drivers worldwide (DMR, 2023).

1. Motivation

Why managing such platforms is difficult?

1. Demand heterogeneous in time and space.

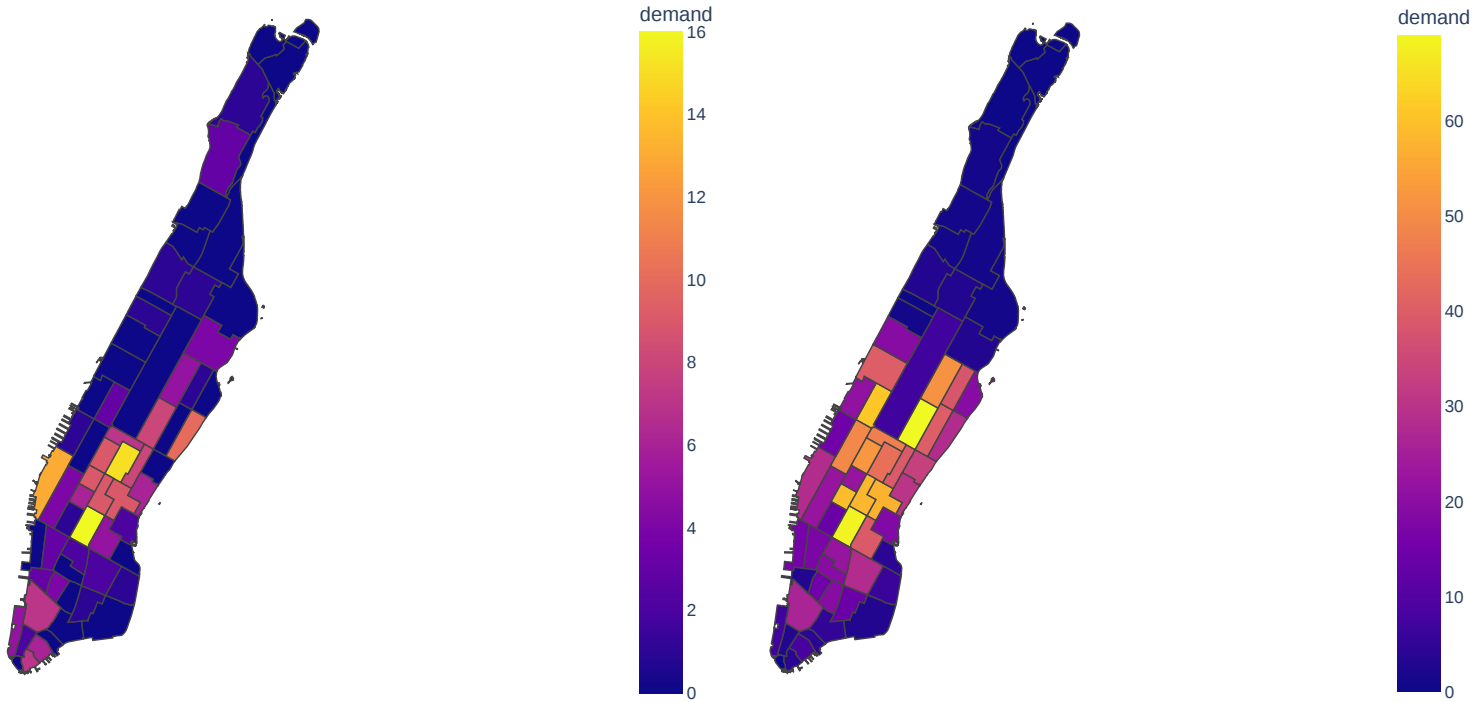


Figure: Realized demand from Midtown Center, distributed per destination between 8:00-8:59 AM (a) and 18:00-18:59 AM (b) on February 6th, 2018.

1. Motivation

Why managing such platforms is difficult? 2. Customer service at one time has a direct impact on the service at subsequent time.

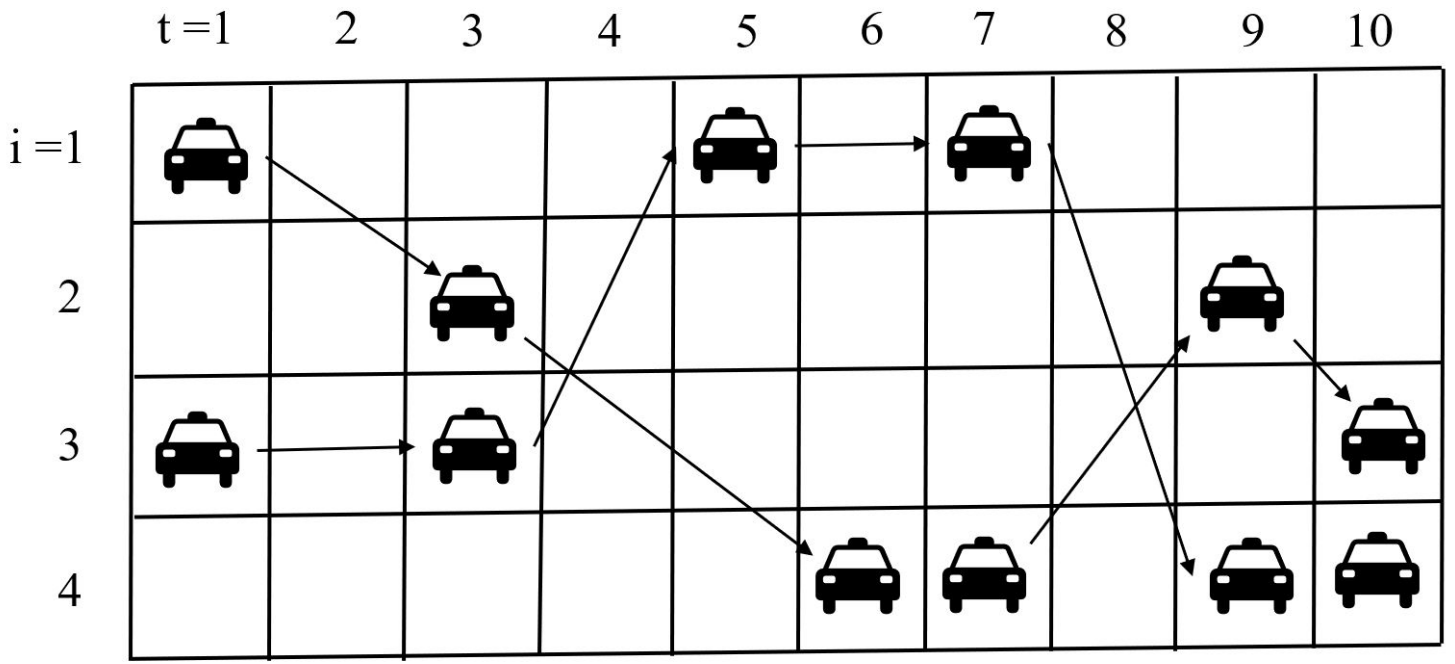


Figure: A sample path for 2 drivers in a network of 4 locations over 10 periods.

1. Motivation

Why managing such platforms is difficult? 3. Drivers and customers exhibit complex behaviors.

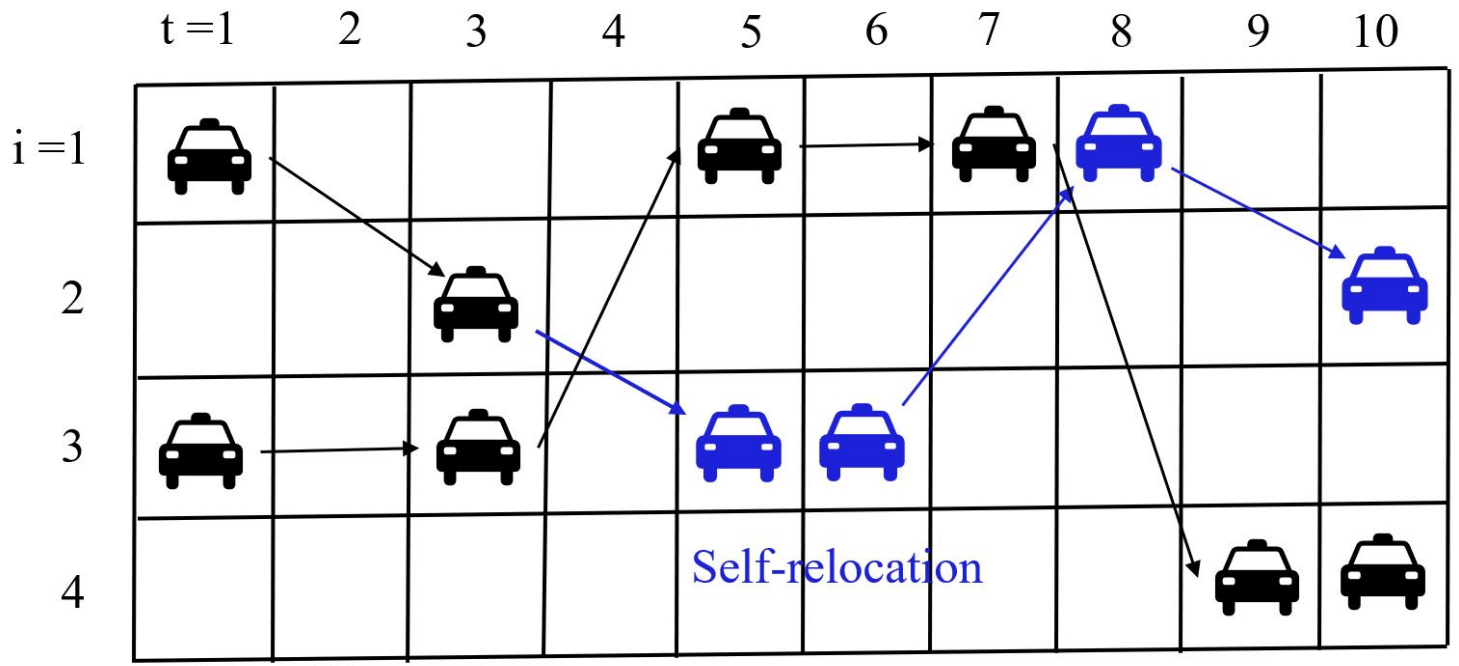


Figure: A sample path for 2 drivers in a network of 4 locations over 10 periods; one driver decides to self-relocates.

1. Motivation

In summary, managing such platforms is difficult due to:

1. Heterogeneous demand in time and space.
2. Current customer service impacts future customer service.
3. Drivers and customers exhibit complex behaviors.

How can we deal with these difficulties?

1. Dynamic pricing (demand and supply).
2. Driver dispatching: matching and repositioning (supply).

2. Problem Setting

We consider

- A set $\mathcal{N} \equiv \{1, ..., N\}$ of locations (e.g., neighborhoods),
- A set $\mathcal{T} \equiv \{0, ..., T\}$ of discrete decision times (e.g., every five minutes).

2. Problem Setting

We consider

- A set $\mathcal{N} \equiv \{1, ..., N\}$ of locations (e.g., neighborhoods),
- A set $\mathcal{T} \equiv \{0, ..., T\}$ of discrete decision times (e.g., every five minutes).

Given

- potential demand rates $\boldsymbol{\Lambda}_t \in \mathbb{R}^{N \times N}$, and
- expected transit times $1/\boldsymbol{\tau} \in \mathbb{R}^{N \times N}$,

2. Problem Setting

We consider

- A set $\mathcal{N} \equiv \{1, ..., N\}$ of locations (e.g., neighborhoods),
- A set $\mathcal{T} \equiv \{0, ..., T\}$ of discrete decision times (e.g., every five minutes).

Given

- potential demand rates $\boldsymbol{\Lambda}_t \in \mathbb{R}^{N \times N}$, and
- expected transit times $1/\boldsymbol{\tau} \in \mathbb{R}^{N \times N}$,

the platform makes

- pricing decisions $\mathbf{p}_t \in [0, 1]^N$, and
- dispatching decisions $\mathbf{d}_t \in \mathbb{Z}^{N \times (N+1)}$.

2. Problem setting

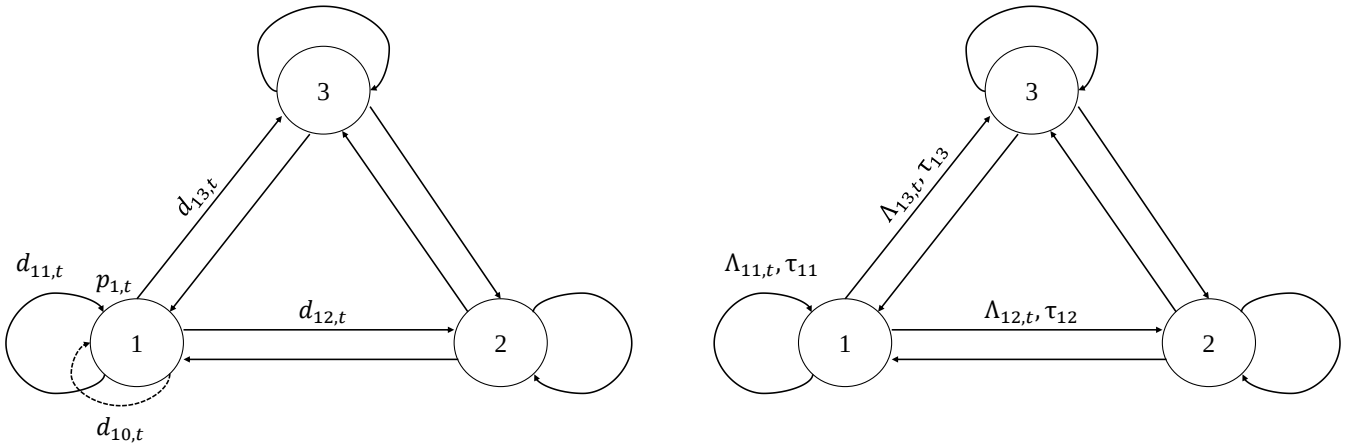


Figure: Schematic illustration of the problem.

2. Problem setting

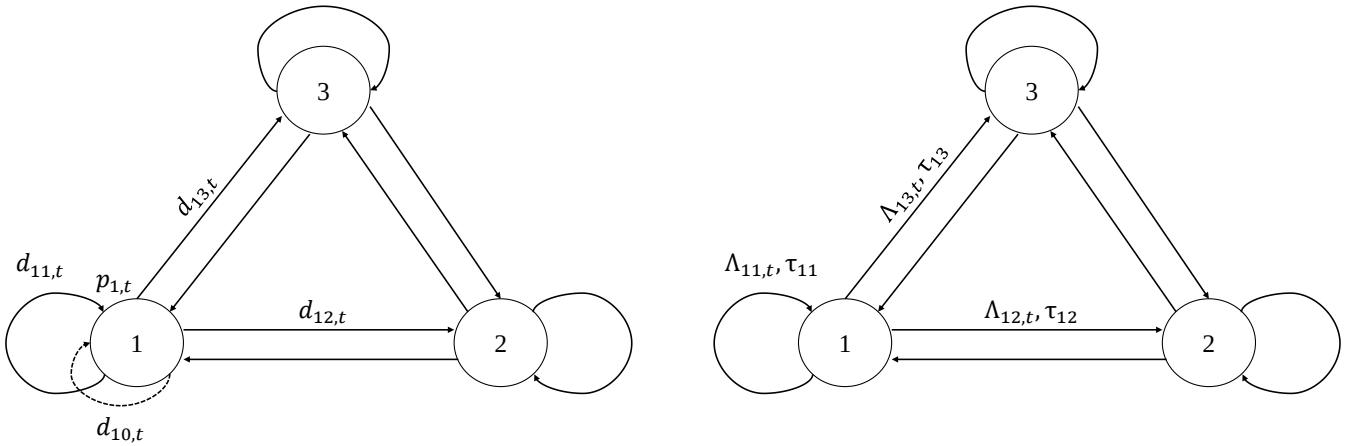


Figure: Schematic illustration of the problem.

Assumptions:

- Transit times are exponentially distributed.
- Drivers can only be assigned to customers in the same location.
- Drivers always accept to be repositioned by the platform.
- Customers opt for an outside option (e.g., subway) if not satisfied within the time period.

2. MDP Formulation

We can formulate this setting as an MDP.

- **State** $\mathbf{s}_t = (\mathbf{x_t}, \mathbf{y}_t)$: Drivers available in each location $\mathbf{x}_t \in \mathbb{Z}^N$, and drivers in transit between locations $\mathbf{y}_t \in \mathbb{Z}^{N \times N}$, such that

$$\sum_{i=1}^{N} x_{i,t} + \sum_{i=1}^{N} \sum_{j=1}^{N} y_{ij,t} = F, \quad t \in \mathcal{T}. \tag{1}$$

- **Decisions** $\mathbf{a}_t = (\mathbf{p}_t, \mathbf{d}_t)$: Prices $\mathbf{p}_t \in [0,1]^N$ and number of drivers left available or dispatched $\mathbf{d}_t \in \mathbb{Z}^{N \times (N+1)}$, such that:

$$\sum_{j=0}^{N} d_{ij,t} = x_{i,t}, \quad , i \in \mathcal{N}, t \in \mathcal{T}. \tag{2}$$

2. MDP Formulation

Random events

- Realized demand $\mathbf{c}_t \in \mathbb{Z}^{N\times N}$ follow Poisson distributions: $\mathbf{c}_t \sim Pois\Big(\boldsymbol{\lambda}_t(\mathbf{p}_t)\Big)$, where $\lambda_{ij,t} = \Lambda_{ij,t}\big(1 - G(p_{i,t})\big) = \Lambda_{ij,t}\big(1 - p_{i,t}\big)$.

- Drivers chooses to self-relocate w.p. β, and chooses location j w.p.

$$\psi(j \mid i,\, t) = \frac{p_{j,t} \cdot \tau_{ij}}{\sum_{k=1}^{N}(p_{k,t} \cdot \tau_{ik})}. \tag{3}$$

- Drivers completing service $\mathbf{z}_t \in \mathbb{Z}^{N\times N}$ follow binomial distributions: $\mathbf{z}_t \sim Bin\Big(\mathbf{y}_t + \mathbf{d}_t + \mathbf{w}_t, 1 - e^{-\boldsymbol{\tau}}\Big)$, with $\mathbf{w}_t$ denoting the self-relocating drivers.

2. MDP Formulation

Transition function

The number of drivers available at decision time $t+1$:

$$x_{i,(t+1)} = d_{i0,\,t} - \sum_{j=1}^{N} w_{ij,t} + \sum_{j=1}^{N} z_{ji,t}, \qquad i \in \mathcal{N},\ t \in \mathcal{T}.$$

The number of drivers in transit at decision time $t+1$:

$$y_{ij,(t+1)} = y_{ij,t} + d_{ij,t} + w_{ij,t} - z_{ij,t}, \qquad i \in \mathcal{N},\ j \in \mathcal{N},\ t \in \mathcal{T}.$$

2. MDP Formulation

Reward function:

$$R_t(\mathbf{s}_t, \mathbf{a}_t) = \sum_{i=1}^{N}\sum_{j=1}^{N} p_{it} \cdot 1/\tau_{ij} \cdot \min(d_{ij,t}, c_{ij,t}) \qquad \textbf{(revenues)}$$

$$- C_r \cdot \sum_{i=1}^{N}\sum_{j=1}^{N} 1/\tau_{ij} \cdot \max(d_{ij,t} - c_{ij,t}, 0) \ \textbf{(repositioning)}$$

$$- C_a \cdot \sum_{i=1}^{N}\sum_{j=1}^{N} \max(c_{ij,t} - d_{ij,t}, 0). \qquad \textbf{(abandonment)}$$

3. Solution Approaches

We want to find an optimal policy $\pi^* = [\pi_0^*, \pi_1^*, ..., \pi_T^*]$ that maximizes

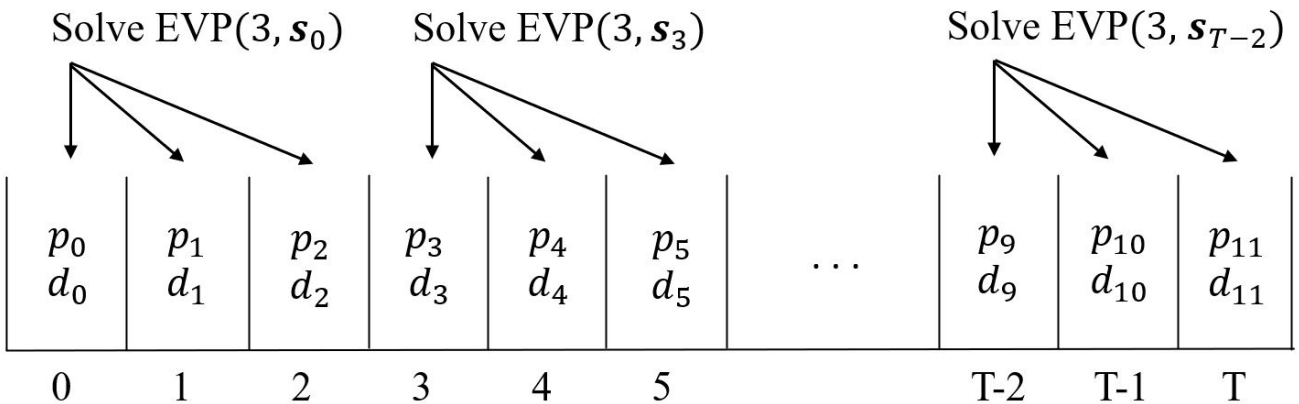
$$\mathbb{E}_\pi \left[\sum_{t=0}^{T} R_t\big(\mathbf{s}_t, \pi_t(\mathbf{s}_t)\big) \,\Big|\, \mathbf{s}_0 \right]. \tag{4}$$

Problems:

- Curse of dimensionality on the state space.
- Curse of dimensionality on the action space.
- Curse of dimensionality on the uncertainty space.

3. Approach #1. Expected Value Program (EVP)

- Idea: Solve a deterministic formulation of the problem (EVP) for the next n decision times, based on the system state $\mathbf{s}_t$.



- Pros: fast, interpretable.
- Cons: ignore stochastic fluctuations, myopic policy.

3. Approach #1. Mathematical Formulation

Let $e_{ij,t}$ be the number of customers served from location i to j at t; let $\zeta_{ij,s}$ be the probability for a driver to complete service from i to j between decision times s and $s+1$.

$$\max \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{t'=t}^{t+n-1} p_{i,t'} \cdot e_{ij,\,t'}/\tau_{ij} - \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{t'=t}^{t+n-1} C^r \cdot \left(d_{ij,\,t'} - e_{ij,\,t'}\right)/\tau_{ij} \tag{5a}$$

$$- \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{t'=t}^{t+n-1} C^a \cdot \left(\lambda_{ij,\,t'} - e_{ij,\,t'}\right) \tag{5b}$$

$$\text{s. t. } e_{ij,\,t'} \le d_{ij,\,t'}, \qquad \forall\, i,\, j,\, t' = t, \ldots, t+n-1, \tag{5c}$$

$$e_{ij,\,t'} \le \lambda_{ij,\,t'}, \qquad \forall\, i,\, j,\, t' = t, \ldots, t+n-1, \tag{5d}$$

$$x_{i,\,t'} \ge \sum_{j=1}^{N} d_{ij,\,t'}, \qquad \forall\, i,\, t' = t, \ldots, t+n-1 \tag{5e}$$

$$x_{i,\,(t'+1)} = x_{i,\,t'} - \sum_{j=1}^{N} d_{ij,\,t'} + \sum_{j=1}^{N}\sum_{s=0}^{t'-t} \zeta_{ji,\,s}\, d_{ji,\,t'-s}$$

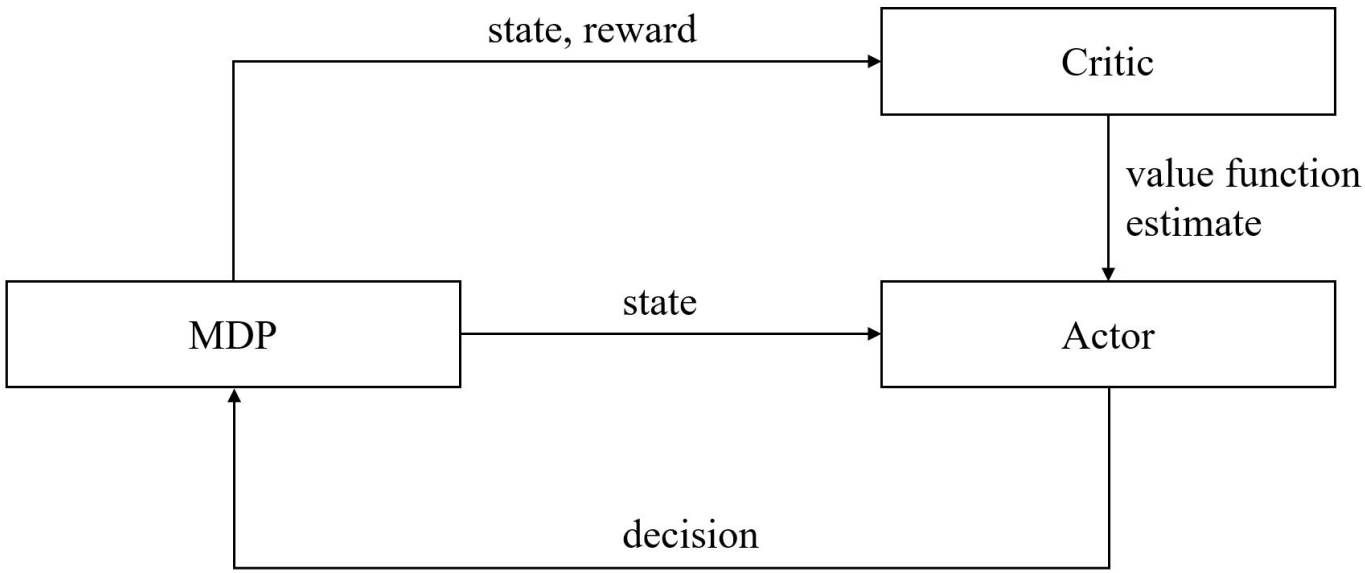
$$+ \sum_{j=1}^{N} \zeta_{ji,\,t'-t}\, y_{ji,\,t}, \quad \forall\, i,\, t' = t, \ldots, t+n-1, \tag{5f}$$

$$\lambda_{ij,\,t} = \Lambda_{ij,\,t} \cdot \left[1 - G(p_{i,\,t})\right], \qquad \forall\, i,\, j,\, t' = t,\, \ldots,\, t+n-1, \tag{5g}$$

$$p_{i,\,t},\ d_{ij,\,t},\ \lambda_{ij,\,t},\ e_{ij,\,t} \ge 0, \qquad \forall\, i,\, j,\, t' = t, \ldots, t+n-1. \tag{5h}$$

3. Approach #2. Actor-Critic DRL

- Idea: Use deep reinforcement learning based on actor-critic algorithms.



- Pros: dynamic policy, powerful approximators.
- Cons: extensive computational efforts, "black-box" solution.

3. Approach #2. Actor-Critic DRL

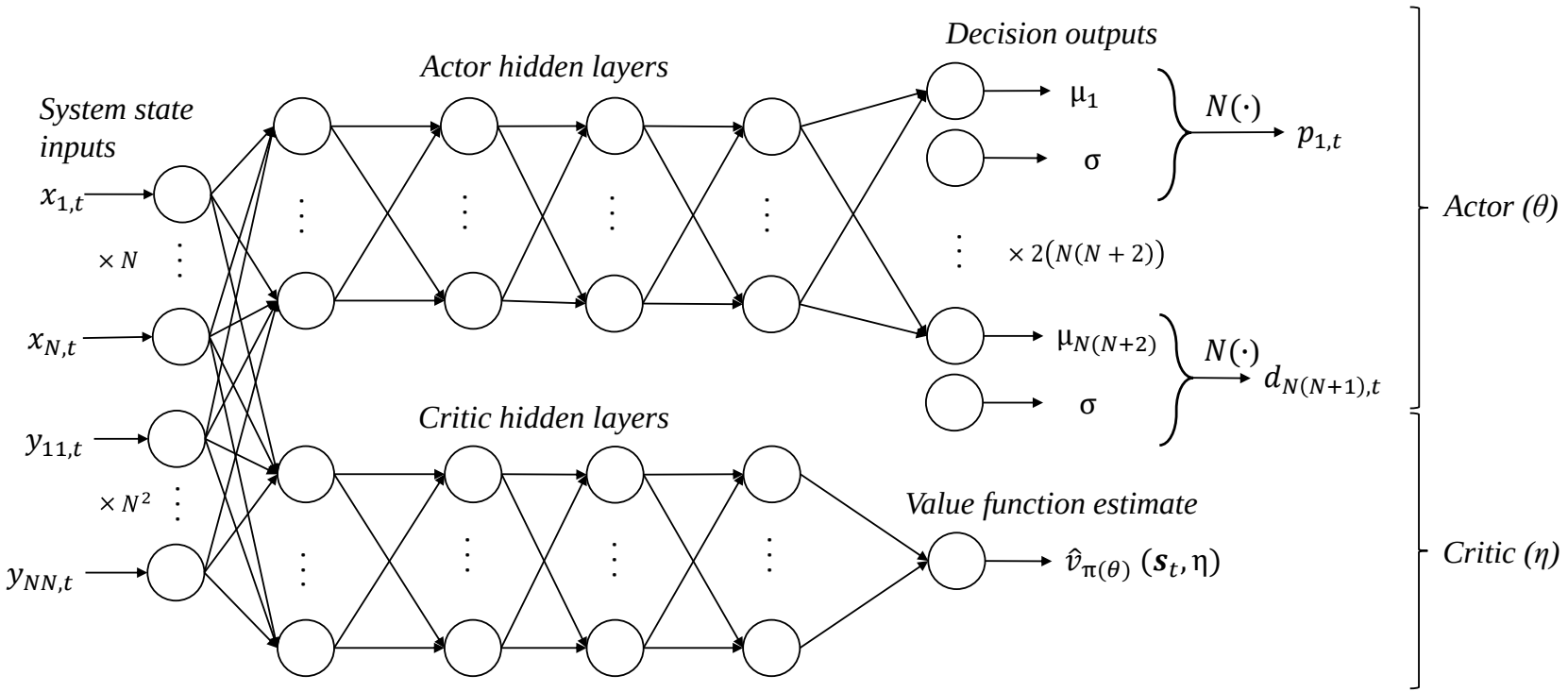


Figure: Architecture of the neural networks

3. Approach #3. Pretrained Actor-Critic DRL

- Idea: We train the actor and critic to reproduce the policy of EVP, then continue learning through usual RL.

Pretraining algorithm

Step 1. Build dataset $\mathcal{D} = \{(s_i, a_i)\}_{i=1}^{I}$ by solving $EVP(n, \mathbf{s}_t)$.

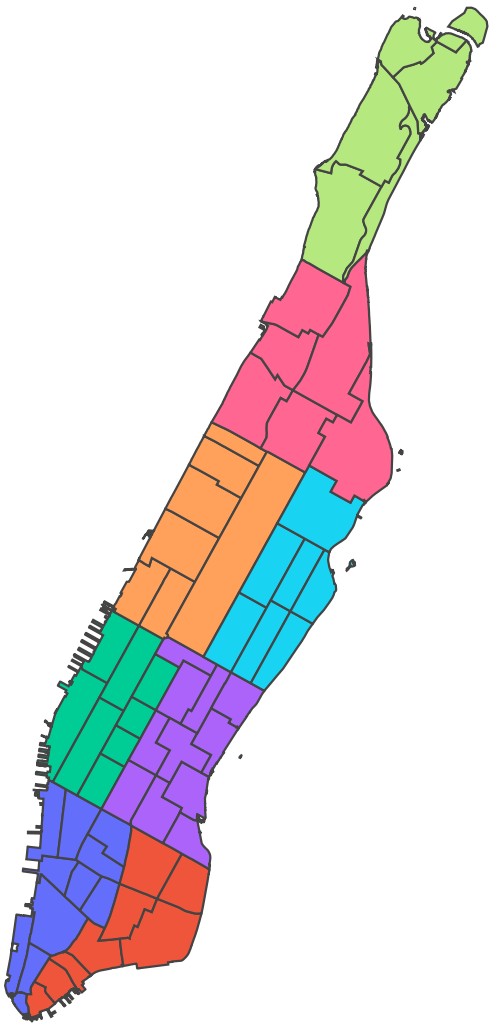
Step 2. For J iterations, minimize difference between $\pi(s_i)$ and a_i, $(s_i, a_i) \in \mathcal{D}$, through stochastic gradient descent (SGD).

Step 3. For K iterations, do:

- Run policy π for L decision times, store total returns G_l, and states $\mathbf{s}_l$.
- Minimize difference between G_l and $\hat{v}(\mathbf{s}_l)$, through SGD.

- Pros: reduced computational efforts, more interpretable, no early stuck in local maxima.

4. Case Study: Manhattan



- We cluster Manhattan into $N = 8$ locations, $T = 24$ periods.
- $F = 1200$ drivers, $\beta = \{0, 0.1\}$, $C_a = 1.0$, $C_r = 0.3$.
- From data of 2019, Feb. 6th, 7:00 a.m to 9:00 a.m, we extract potential demand rates $\Lambda_{ij,t}$ and mean transit times $1/\tau_{ij}$ (5-min intervals).

4. Performance of Solution Approaches

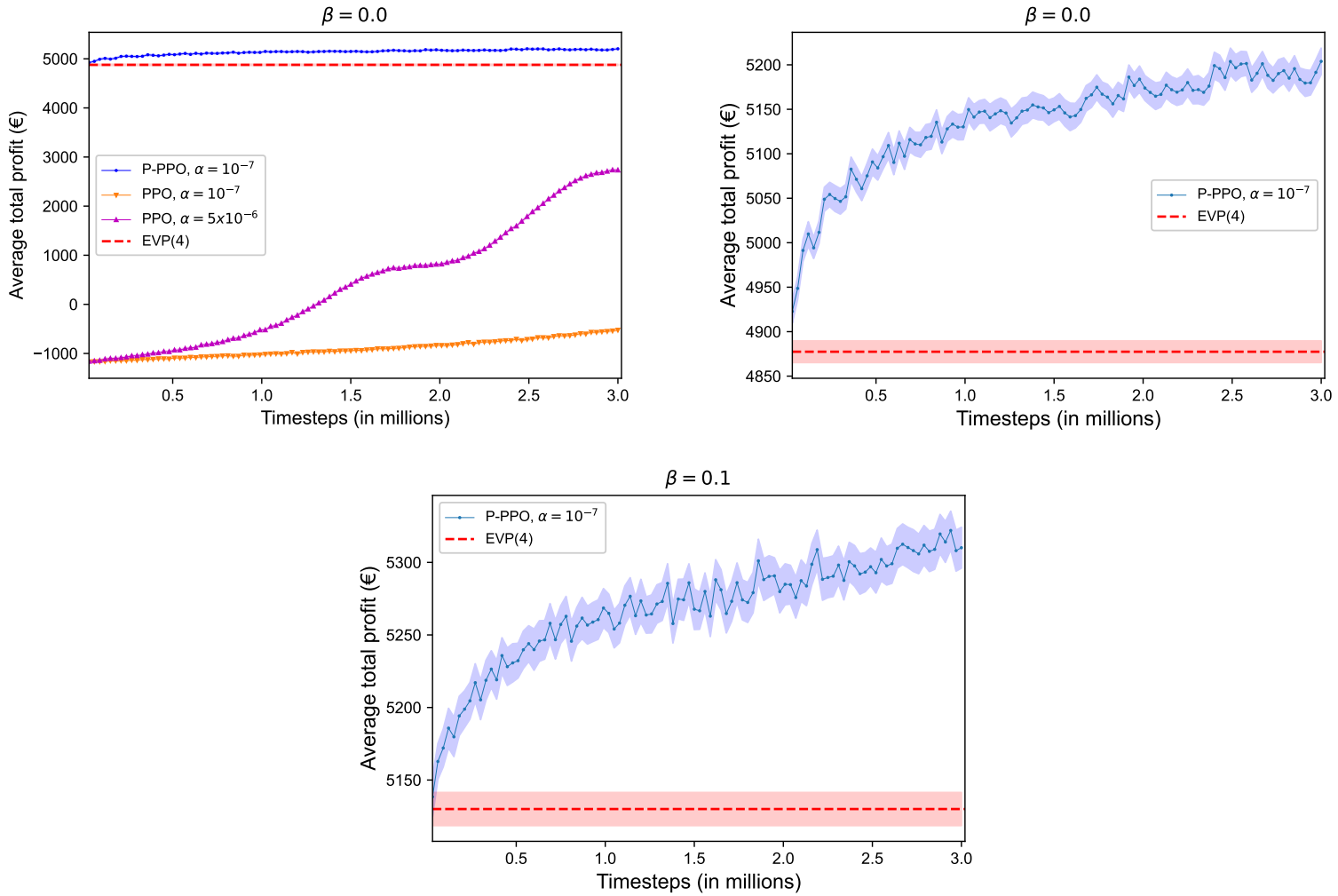


Figure: Training curves for P-PPO, and PPO, in comparison with *EVP*(4).

4. Performance of Solution Approaches

Instance	CPU time (hours)		Final performance (€)		
	PPO	P-PPO	EVP(4)	PPO	P-PPO
$\overline{\Lambda} = 0.75$, $\overline{\tau} = 0.75$	75.52	24	6534	−246	6639(1.6%)
$\overline{\mathbf{\Lambda}} = \mathbf{1.0}$, $\overline{\tau} = \mathbf{1.0}$	73.65	21.51	9141	4496	9474(3.6%)
$\overline{\Lambda} = 1.25$, $\overline{\tau} = 1.25$	71.04	13.26	10232	6535	10939(6.9%)
$F = 800$	71.04	24	7848	3480	8192(4.4%)
$\mathbf{F = 1200}$	73.65	21.51	9141	4496	9474(3.6%)
$F = 1600$	90	13.48	9054	2043	9195(1.6%)
$\beta = 0.0$	73.62	20.39	8460	4340	8874(4.89%)
$\beta = \mathbf{0.1}$	73.65	21.51	9141	4496	9474(3.6%)
$\beta = 0.2$	76.55	24	9042	4653	9489(4.9%)
$\beta = 0.3$	90	24	8889	4800	9340(5.1%)
$\beta = 0.4$	90	24	8743	4935	9103(4.1%)
$C_a = 0.0$	69.08	24	9605	2926	9824(2.2%)
$C_a = 0.25$	73.82	16.57	9361	3874	9635(2.9%)
$\mathbf{C_a = 0.5}$	73.65	21.51	9141	4496	9474(3.6%)
$C_a = 0.75$	88.06	22.94	8906	4754	9231(3.5%)
$C_a = 1.0$	87.48	24	8695	4355	9028(3.8%)
$C_r = 0.0$	90	24	11099	10978	11070(0.0%)
$C_r = 0.25$	90	24	9769	7707	10041(2.8%)
$\mathbf{C_r = 0.5}$	73.65	21.51	9141	4496	9474(3.6%)
$C_r = 0.75$	75.45	19.72	8539	1319	8769(2.7%)
$C_r = 1.0$	75.52	18.24	7937	−4404	8290(4.4%)

Table: Performances of *EVP*(4), *PPO*, and pretrained *PPO*, in terms of computaton time, final expected profit for instances with $N = 4$, $T = 36$. The final performances are averaged over 300 episodes.

4. Policy Analysis

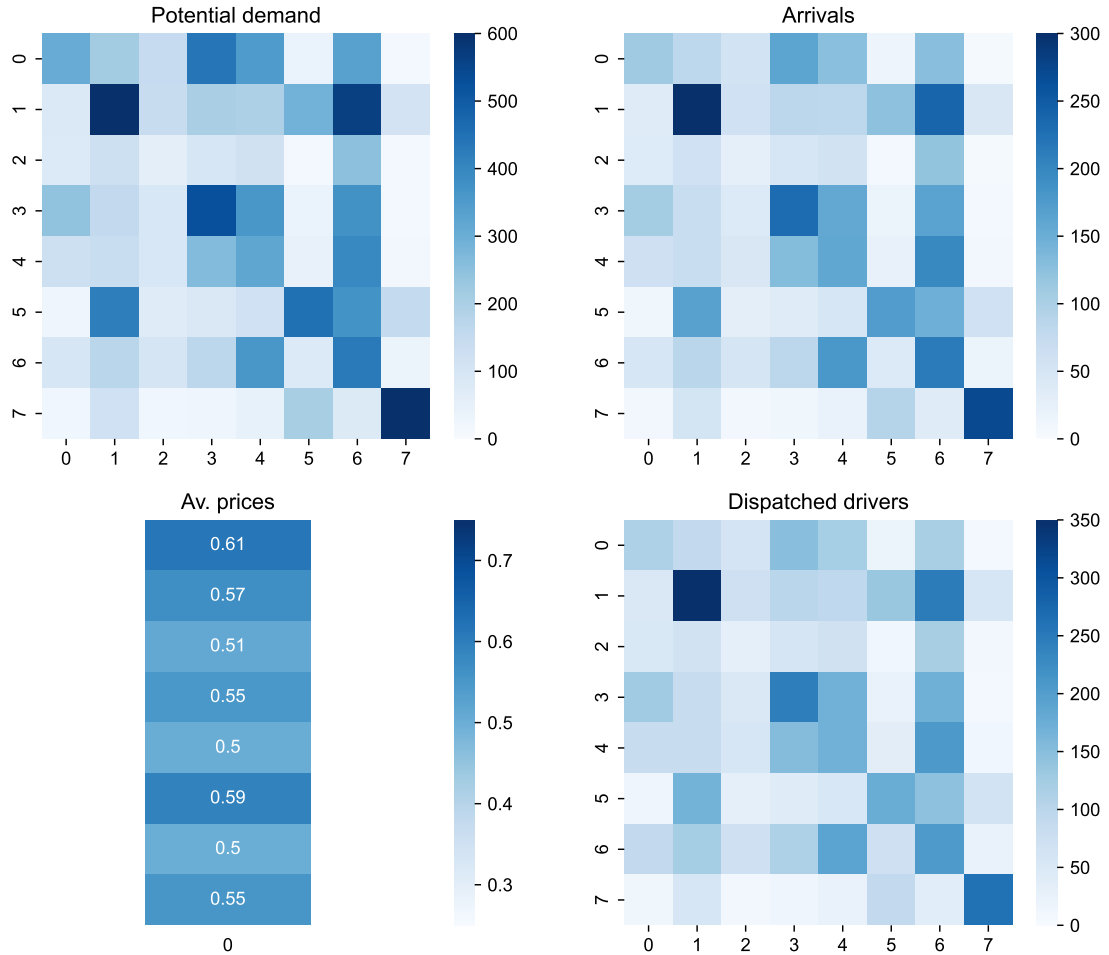


Figure: Heat maps of the total potential demand rates, total arrivals, average prices and total dispatched drivers. The results are averaged over 300 episodes.

4. Policy Analysis

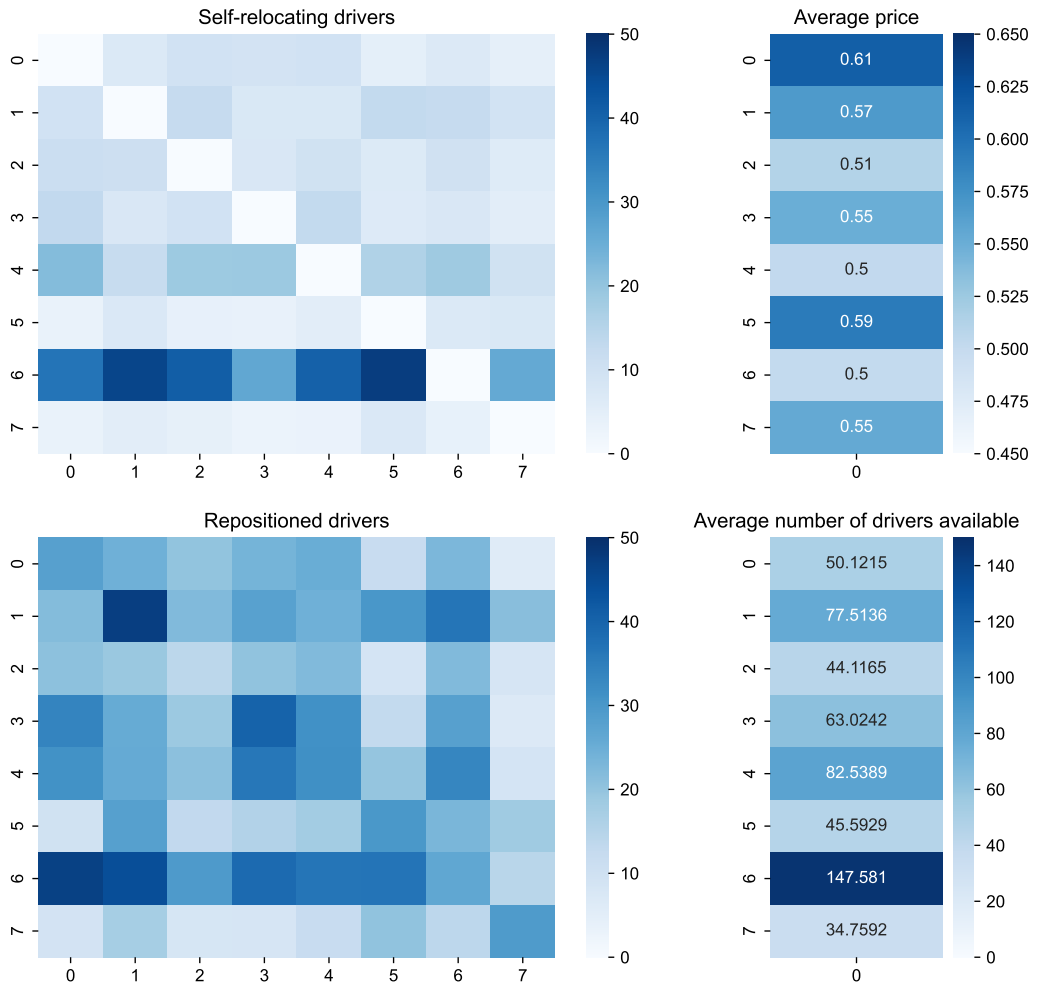


Figure: Heat maps of the total self-relocating drivers, drivers repositioned by the platform, average prices and average number of available drivers.

4. Policy Analysis

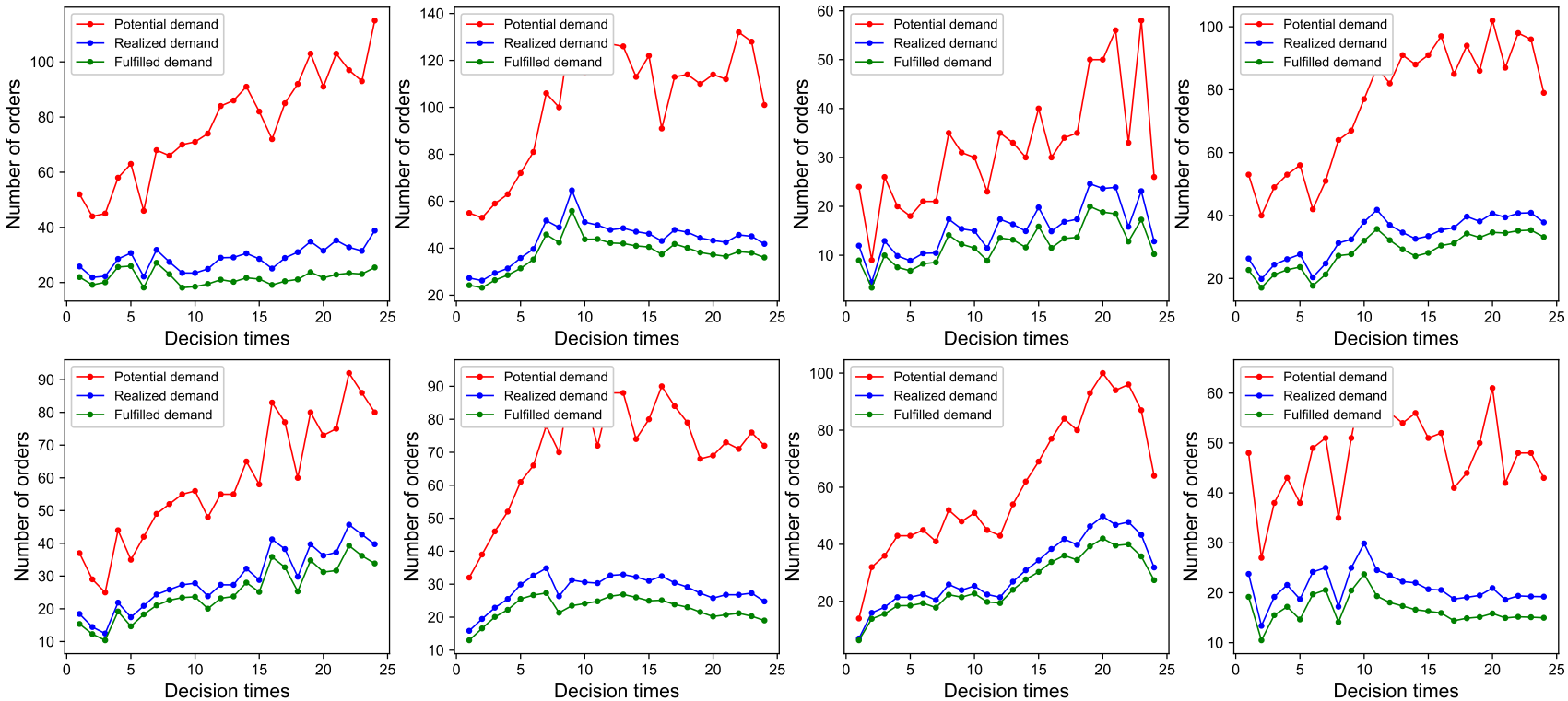


Figure: Evolution of the gaps between the potential demand, the realized demand and the fulfilled demand in each location.

4. Policy Analysis

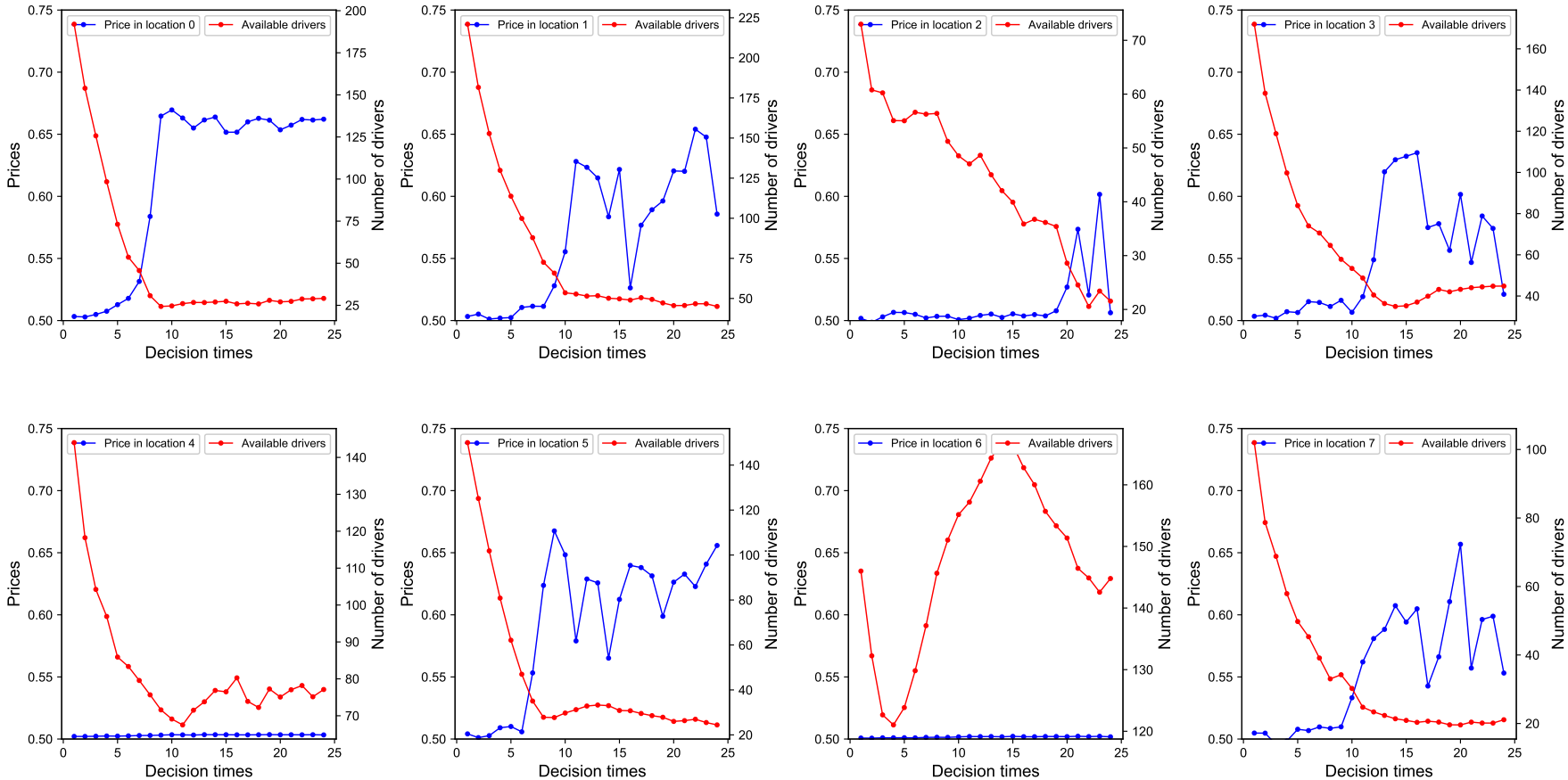


Figure: Evolution of the price and the number of available drivers in each location.

4. Policy Robustness

We study the performance of pretrained PPO, when applied to new data (Feb, 13th).

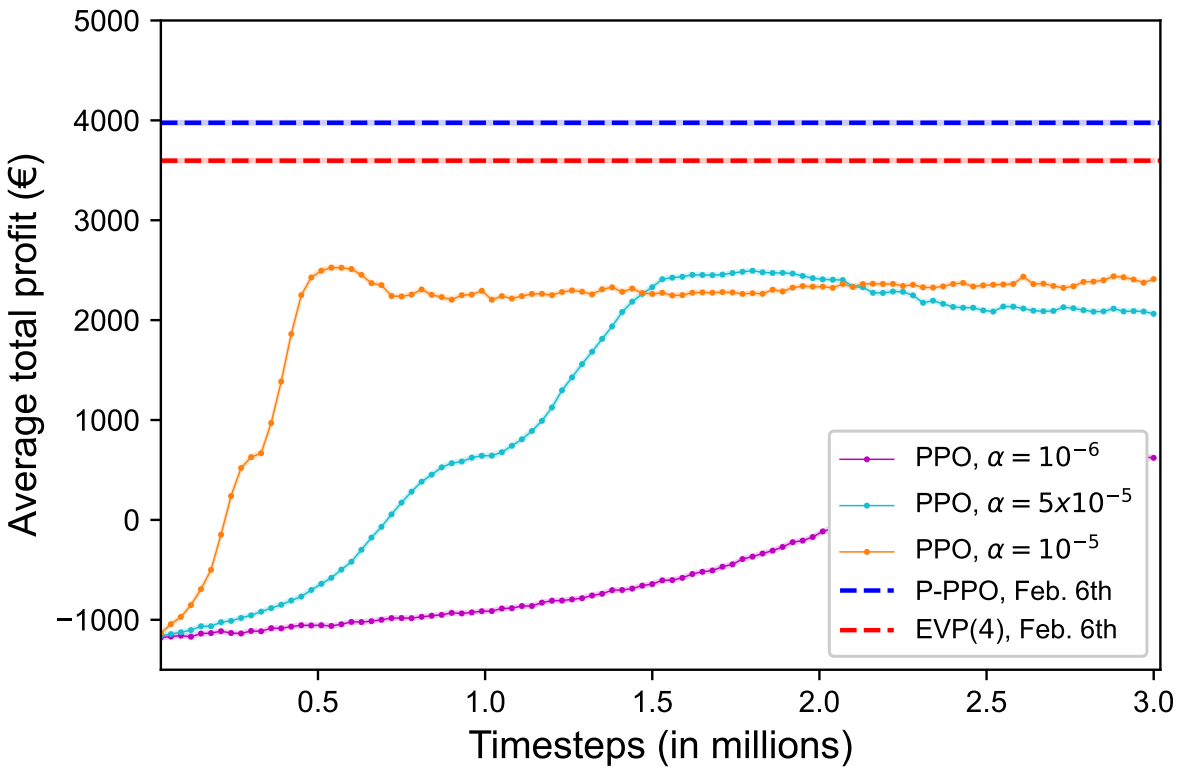


Figure: Performance comparison of the policies when applied to data of the next week (Feb. 13th).

4. Policy Robustness

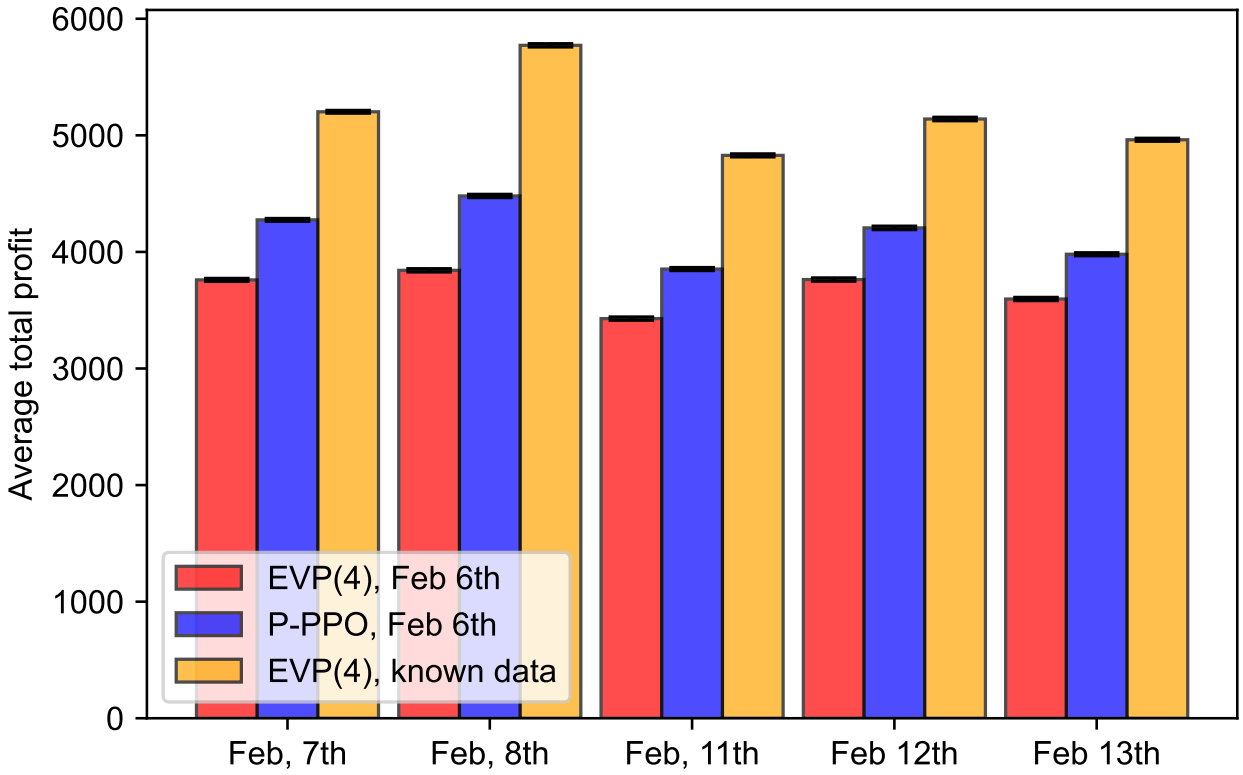


Figure: Performance comparison of P-PPO with $EVP(4)$, when applied to data of the next 5 days.

5. Conclusion

Summary:

- Markovian model for joint pricing and dispatching decisions.
  1. Spatial-temporal imbalances.
  2. Stochastic demand and transit times.
  3. Customer and driver behavior.
- Approaches based on mathematical programming with DRL.
  1. Expected value program.
  2. Actor-critic DRL.
  3. Neural network pretraining.

Future steps:

- Training a more robust policy.
- More elaborated self-relocating behavior of the drivers.
- Multistage stochastic optimization.