Topics in Reinforcement Learning

Tabular RL Project: Finite Inventory Pricing

Team 7

Himanshu Singh Vikaskumar Kalsariya

March 11, 2025

Overview

Introduction 3

Experiment 7

Analysis 16

Introduction: Problem Statement

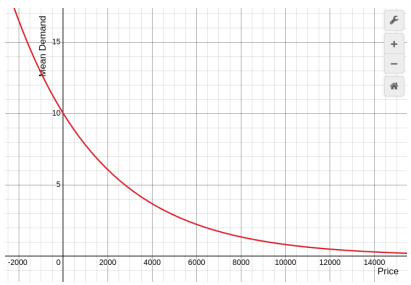
- There are K seats with a booking window of T days.
- Design a policy, $p_t, t \in \{1, 2, \ldots, T\}$ that will maximize the revenue over the booking window.
- Demand for seats at price p_t, denoted by $d(p_t)$, is a random variable supported on positive integers.
- Use RL to learn the optimal policy in case demand function is unknown.

Introduction: Demand Function

- On any given day, we want to sample an order for a number of seats. The total order can be represented as a non-negative integer.
- The demand function should ideally depend on a positive real-valued price and a source of randomness.

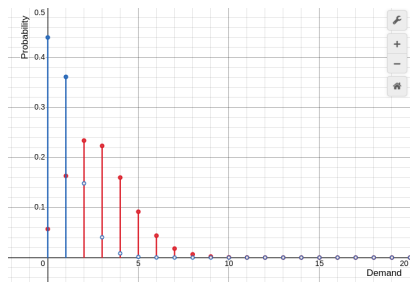
$$d : \mathbb{R}^+ \times U \to \mathbb{Z}^+ \cup \{0\}$$

- One way to capture the dependence on price is by changing the mean of the distribution of demand function as a function of the price.



(a) Mean Demand at a Price

$\lambda(p) = 10e^{-0.00025x}$



(b) Demand Distribution

$d(p) = poisson(\lambda(p))$

$p = 5000(\text{red}), 10000(\text{blue})$

Figure: Demand Function

Introduction: MDP Formulation

- State Space: Number of seats available for sale.

$$\mathcal{S} = \{0, 1, \ldots K\}$$

- Action Space: Price of a seat on current day.

$$\mathcal{A} = \{p_{min}, p_{min} + \Delta, \ldots, p_{min} + n \cdot \Delta\}$$

- Transition: Number of seats left after sale on current day.

$$s'(s, p) = s - min(s, d(p))$$

- Reward: Price of all seats sold on the current day.

$$r(s, p, s') = p \cdot (s - s')$$

Introduction: MDP Formulation

- Time Horizon: Booking window of T days for price adjustment and sales.

$$\mathcal{T} = \{0, 1, \ldots T\}$$

- Absorbing/Terminal States: No sales can be made after the end of booking window or seats in inventory.

$$V_0(s) = 0, \forall s \in \mathcal{S}$$

$$V_t(0) = 0, \forall t \in \mathcal{T}$$

where $V_t(s)$ represents the value function when we have s seats in inventory and have t days left in the booking window.

Experiment: Adapting Algorithms to Finite Horizon

- Recall that we already know $\{V_0(s)\}_{s\in\mathcal{S}}$. We can use backtracking for evaluating any policy (Policy Iteration), or solve for the optimality operator (Value Iteration).

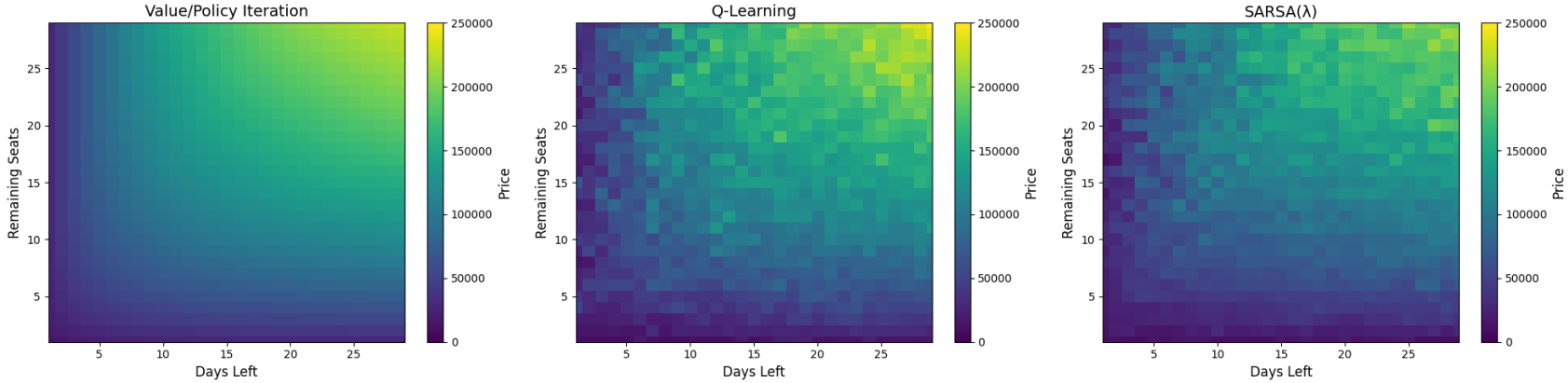
$$V_{t+1}(s) = \sum_{p\in\mathcal{A}} \pi(p|s) \Bigg[\sum_{s'>0\in\mathcal{S}} P(d = s - s' \mid p)\ \{r(s,p,s') + \gamma V_t(s')\} + P(d \geq s \mid p)\,\{r(s,p,0) + \gamma V_t(0)\} \Bigg]$$

- For Q-Learning and SARSA(λ), we sample not just the initial state but also the initial timestep for generating the episodes. The timestep decreases with each transition, terminating in one of the absorbing states $\{V_0(s)\}_{s\in\mathcal{S}}$.
- For the step size, we use $\alpha = \left\lceil \frac{10}{m+1} \right\rceil$, borrowing from the paper Finite Horizon Q-learning by Vivek and Shalabh.

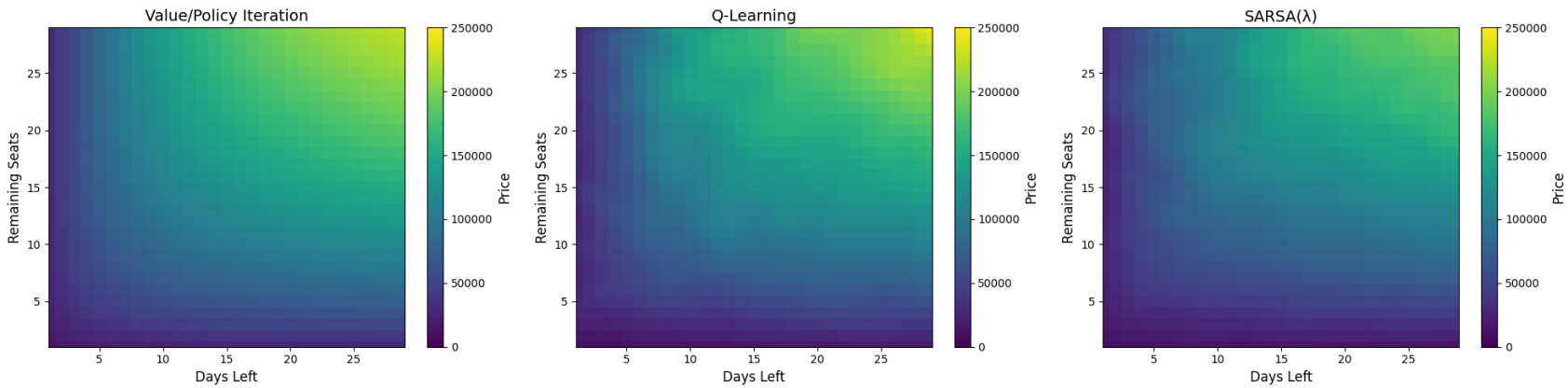
Experiment: Hyperparameters

Hyperparameter	Value
Inventory (#Seats)	30
Booking Window (#Days)	30
Prices	$\{5000, 7500, 10000, 12500, 15000\}$
Discount Factor	0.99
Epsilon	1.00
Trace Decay	0.90
Number of Episodes	100,000

Experiment: Value Function



(a) Original Values

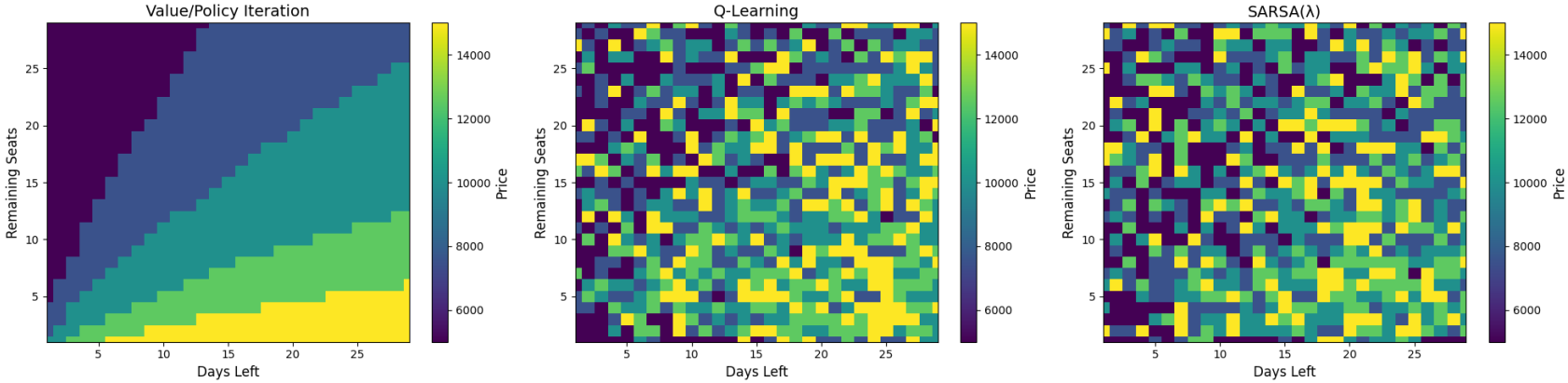


(b) Gaussian Filtered ($\sigma = 1$) [a]

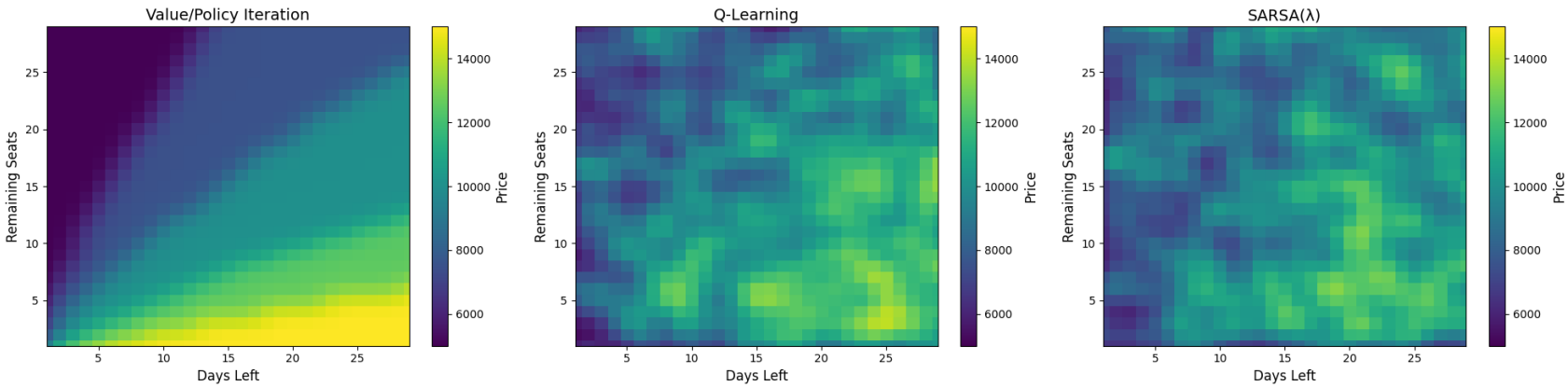
[a] Equivalent to averaging in a 3×3 window.

Figure: Value function learned by different algorithms.

Experiment: Learned Policies



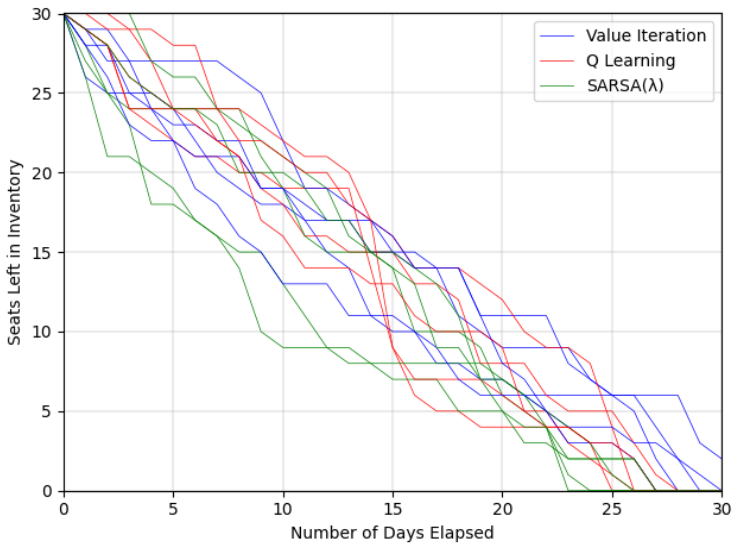
(a) Original Policy



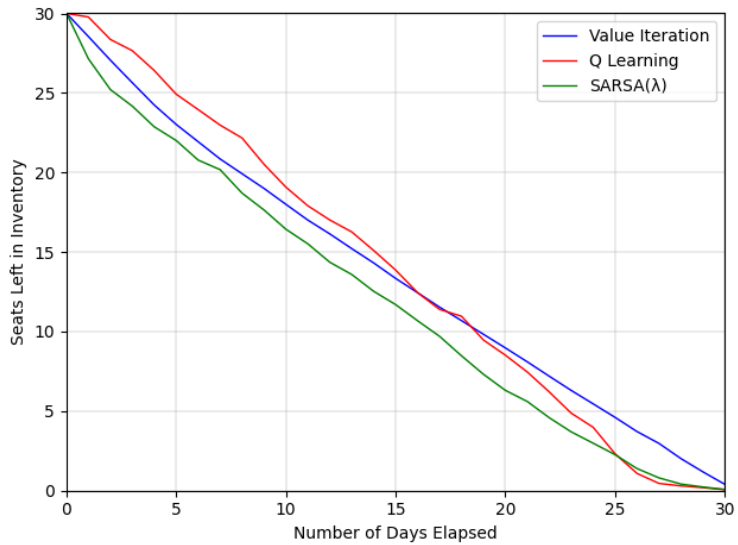
(b) Gaussian Filtered ($\sigma = 1$)

Figure: Policy learned by different algorithms.

Experiment: Simulation of Different Policies



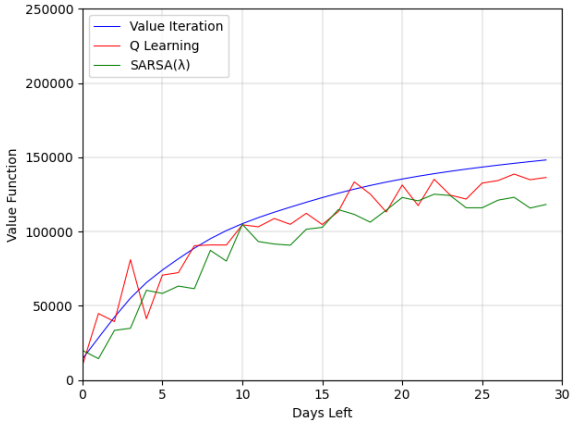
(a) Individual Trajectories



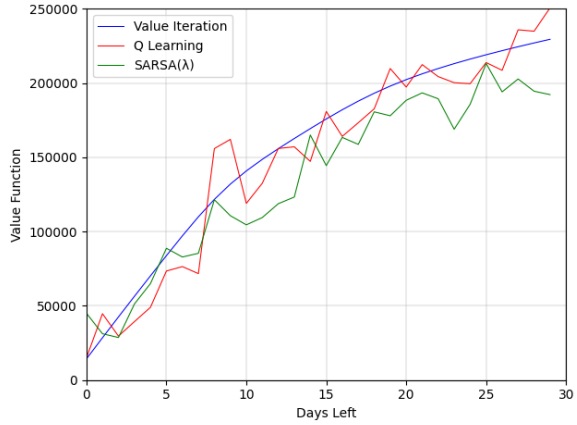
(b) Average Trajectory

Figure: Trajectory of independent simulations.

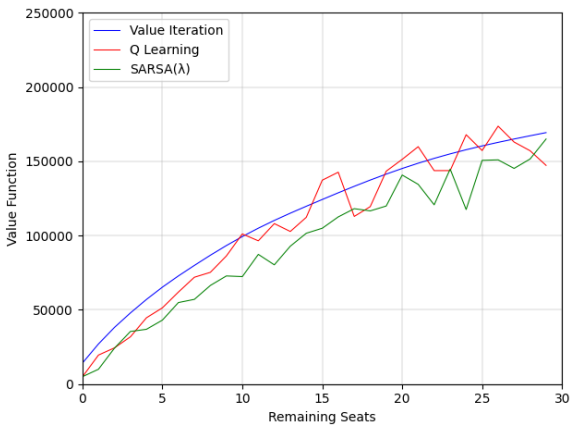
Experiment: Trends in Value Function



(a) Fixed Inventory (k=15)



(b) Fixed Inventory (k=30)

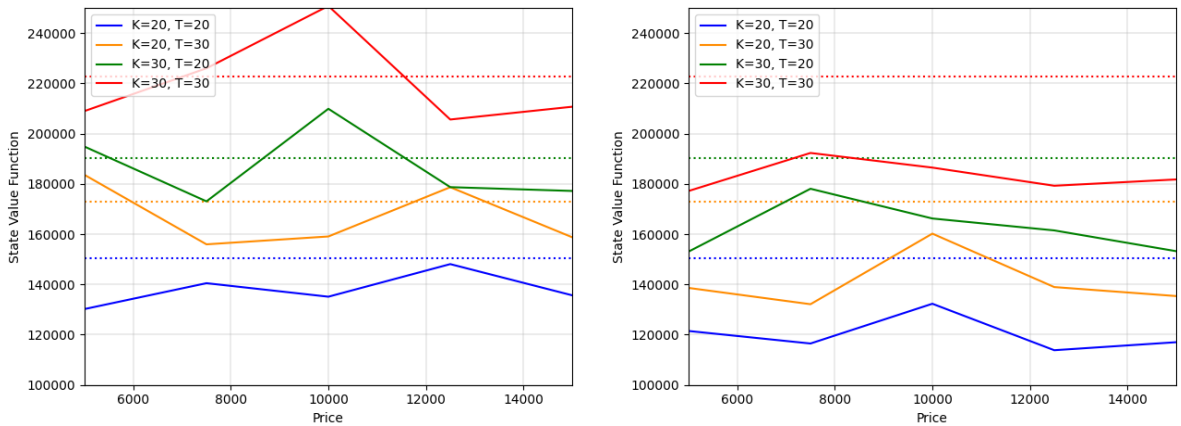


(c) Fixed Timestep (t=15)

(d) Fixed Timestep (t=30)

Figure: Variation in value function with inventory and timestep.

Experiment: Trends in State Value Function

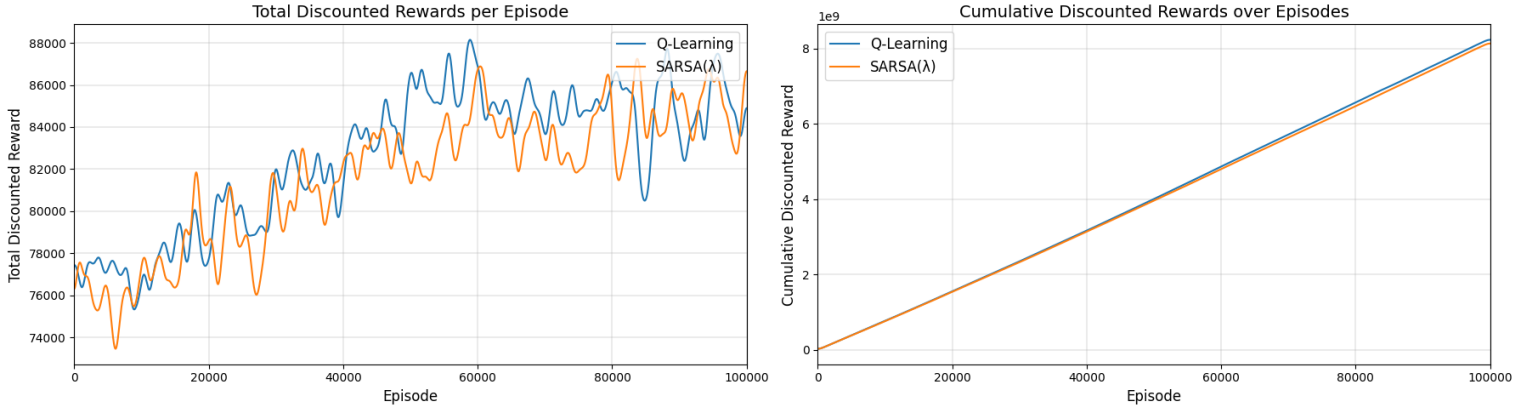


(a) Q-Learning

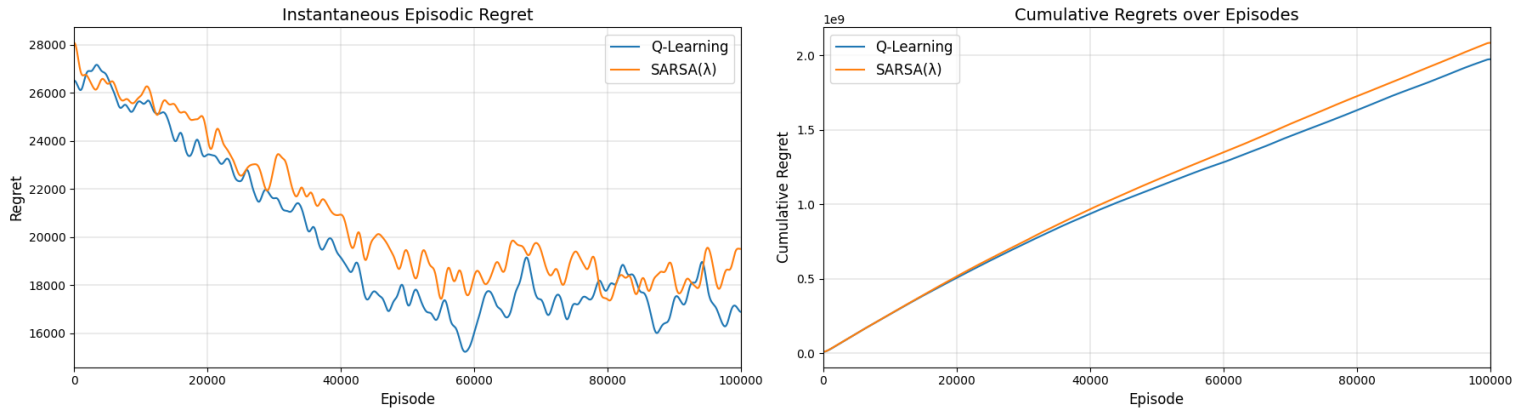
(b) SARSA(λ)

Figure: Variation in state value function for different prices. The dotted line represents the true value function (learned by VI/PI) for the optimal price.

Experiment: Evolution of Rewards and Regrets



(a) Episodic and Cumulative Rewards

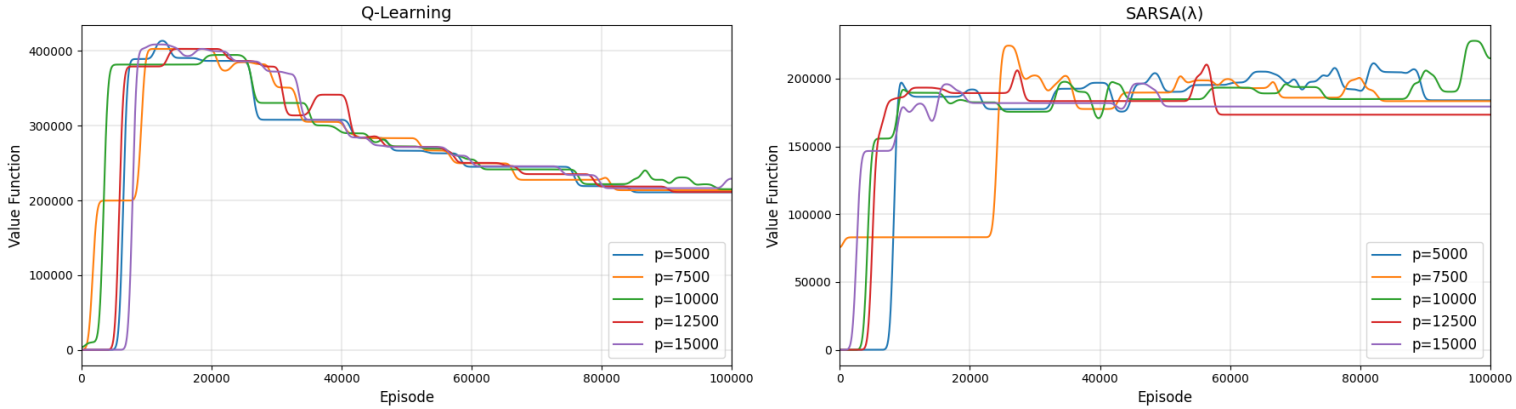


(b) Episodic and Cumulative Regrets [a]

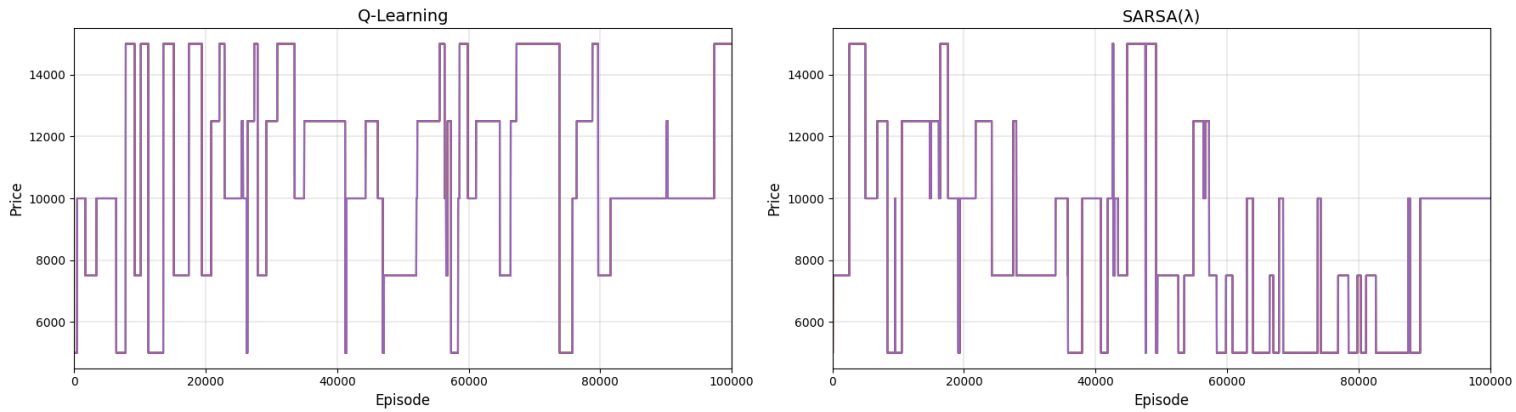
[a] All episodic metrics in the presentation are Gaussian Filtered with $\sigma = 500$, equivalent to averaging over 2000 entries.

Figure: Rewards and regrets incurred by different algorithms.

Experiment: Evolution of Policy and Value Function



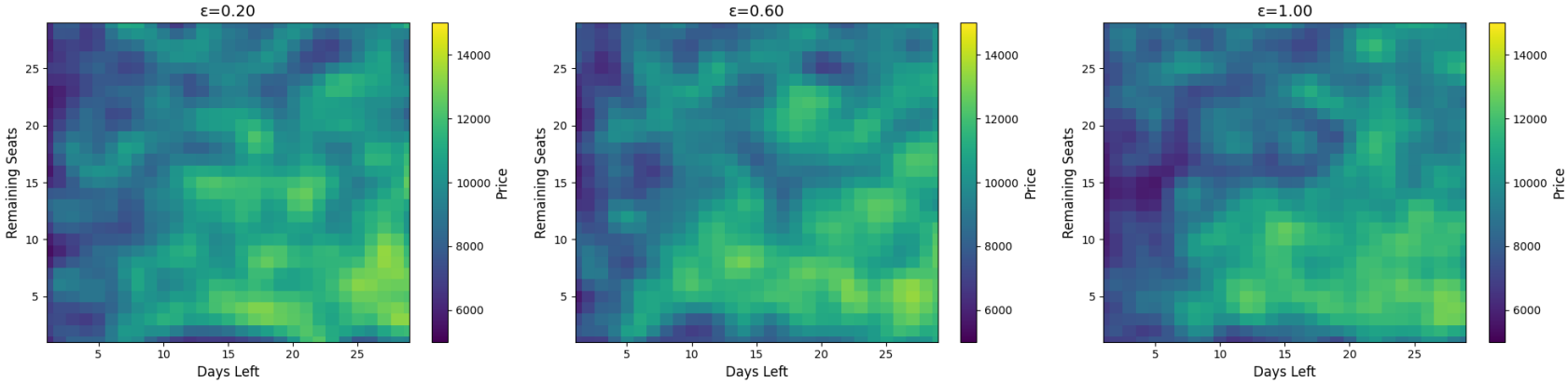
(a) State Value Function



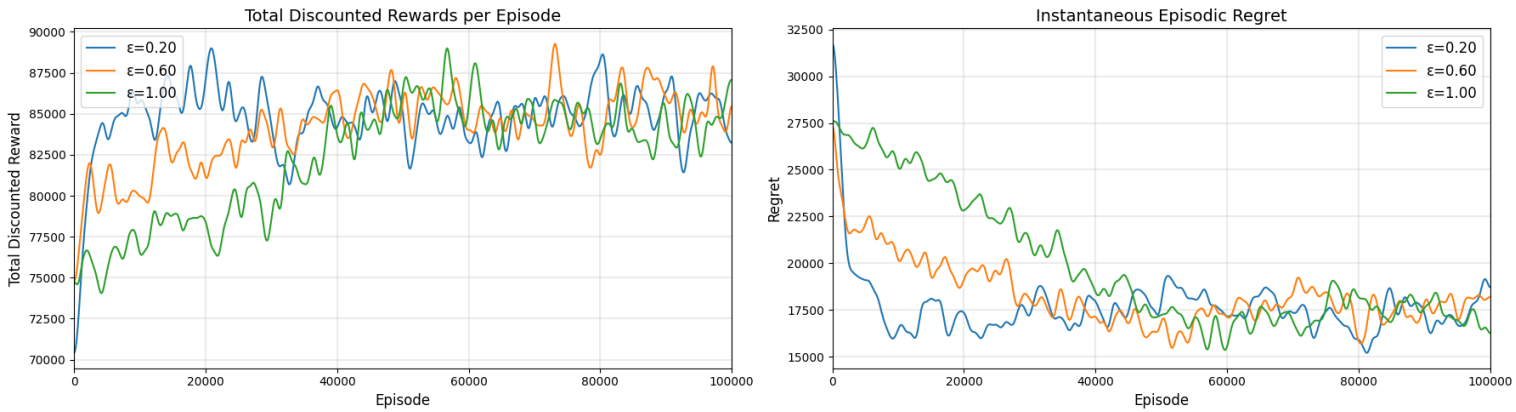
(b) Learned Policy

Figure: Evolution of policy and state value function for a fixed state and time (k=30, t=30).

Analysis: Effect of Exploration



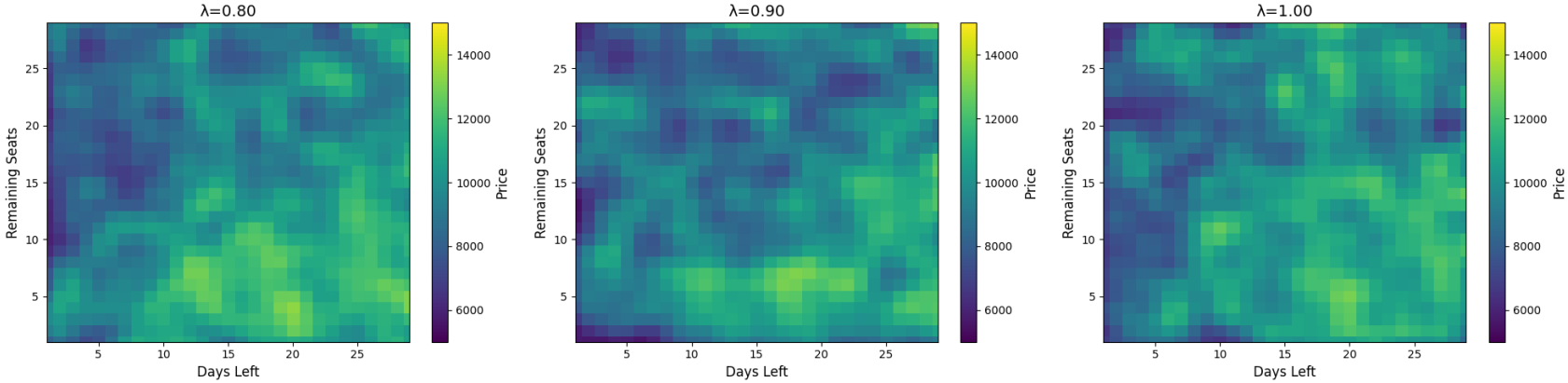
(a) Learned Policy



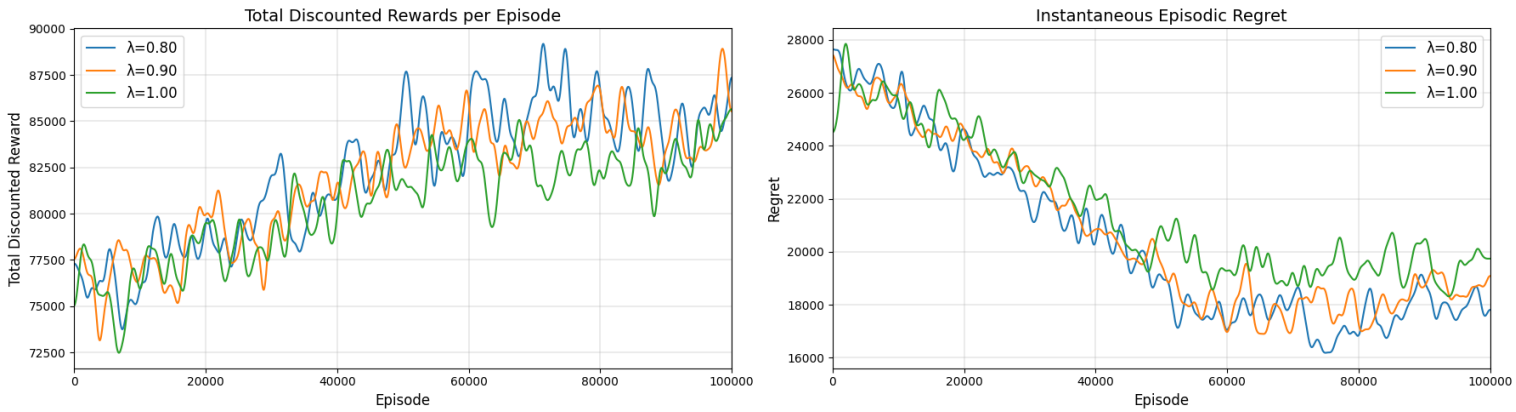
(b) Episodic Rewards and Regrets

Figure: Q-Learning at different levels of $\epsilon-$greedy Exploration

Analysis: Effect of Trace Decay



(a) Learned Policy



(b) Episodic Rewards and Regrets

Figure: SARSA(λ) at different levels of Trace Decay

Conclusion

- Learned policies tend to clear inventory faster than optimal policy, when nearing the end of booking window.
- Learned policies lacked similarity in close neighbourhood, likely due to the possibility of covering up for initial choices later.
- Convergence rate was similar in Q-Learning and SARSA(λ). However, Q-Learning got closer to the optimal policy and acquired less regret.
- Q-Learning tends to overestimate the value function initially, slowly stabilising as the number of episodes increases.
- The effect of exploration and trace decay parameter is not very prominent in the value function, but tends to influence the policy slightly.