Journal of Revenue and Pricing Management (2025) 24:584–600
https://doi.org/10.1057/s41272-025-00519-8

RESEARCH

# Reinforcement learning versus data-driven dynamic programming: a comparison for finite horizon dynamic pricing markets

Fabian Lange[1] · Leonard Dreessen[1] · Rainer Schlosser[1]

Received: 8 October 2024 / Accepted: 7 January 2025 / Published online: 4 February 2025

**Abstract**
Revenue management (RM) plays a vital role to optimize sales processes in real-life applications under incomplete information. The prediction of consumer demand and the anticipation of price reactions of competitors became key factors in RM to be able to apply classical dynamic programming (DP) methods for expected long-term reward maximization. Modern model-free deep Reinforcement Learning (RL) approaches are able to derive optimized policies without explicit estimations of underlying model dynamics. However, RL algorithms typically require either vast amounts of training data or a suitable synthetic model to be trained on. As existing studies focus on one group of algorithms only, the relation between established DP approaches and new RL techniques is opaque. To address this issue, in this paper, we use a dynamic pricing framework for an airline ticket market to compare state-of-the-art RL algorithms and data-driven versions of classic DP methods regarding (i) performance and (ii) required data to each other. For the DP techniques, we use estimations of market dynamics to be able to compare their performance and data consumption against RL methods. The numerical results of our experiments, which include monopoly as well as duopoly markets, allow to study how the different approaches' performances relate to each other in exemplary settings. In both setups, we find that with few data (about 10 episodes) fitted DP methods were highly competitive; with medium amounts of data (about 100 episodes) DP methods got outperformed by RL, where PPO provided the best results. Given large amounts of training data (about 1000 episodes), the best RL algorithms, i.e., TD3, DDPG, PPO, and SAC, performed similarly achieving about 90% and more of the optimal solution.

**Keywords** Dynamic pricing · Decision support · Method comparison · Approximate dynamic programming · Reinforcement learning

## Introduction

### Opportunities of reinforcement learning in revenue management

After the deregulation of the airline industry in the 1970 s, airlines were allowed to select their preferred routes and set their own fares. This was the starting point for a fierce competition in the airline industry, in which every firm made efforts to gain a competitive edge in the market. As a result, airlines introduced advanced optimization techniques to manage their operations. Part of these optimization efforts is the key objective to maximize revenue generated from ticket sales for each flight. A concept commonly referred to as RM in the literature. After improvements of 2–8%, see Smith et al. (1992), in revenue by application of those techniques have been documented, so-called Revenue Management Systems (RMS) soon gained a lot of attention in the airline industry, see also Guerrini et al. (2023).

Part of RMS is to predict the consumer demand in advance in order to adjust the pricing strategy accordingly. However, in many markets, e.g., in e-commerce, such market dynamics are complex and change continuously. In this context, Bondoux et al. (2020) have identified the following challenges that pricing analysts face when implementing RMS: (i) RMS operate on the assumption that the actual customer behavior adheres to a parametric demand model

✉ Rainer Schlosser
rainer.schlosser@hpi.de

Fabian Lange
fabian.lange@student.hpi.de

Leonard Dreessen
leonard.dreessen@student.hpi.de

1 Hasso Plattner Institute, University of Potsdam, Potsdam, Germany

predetermined by the analyst. (ii) Common RMS aim to provide short-term revenue-maximizing prices without exploring alternative strategies to validate its assumptions. (iii) Usually, RMS do not explicitly consider competition.

While forecast-first-then-optimize solution techniques to solve dynamic RM problems such as classical dynamic programming-based approaches are established and well-understood, it is also known that they have their limitations regarding the problem size (cf. the curse of dimensionality) and the underlying assumption of having full information of the dynamic problem, i.e., regarding demand/reward probabilities and state transitions—information that is typically estimated from data using standard regression techniques.

With the rise of AI methods and computational advancements, classes of model-free RL methods have become an interesting alternative to solve dynamic RM problems as they—in contrast to DP methods—are applicable to highly complex problems and they do not require full knowledge of the problem dynamics. RL algorithms can learn an optimized policy by its interactions with a certain environment by balancing a mix of exploration and exploitation. The learning efficiency and the high performance of trained policies have been shown in various domains including revenue management and pricing applications, see Sect. 2.

### Challenges of reinforcement learning in revenue management

Besides the opportunities of automated decision support and revenue maximization under incomplete knowledge, the application of RL agents also comes with different challenges (Kumari and Kumar 2024; Rolf et al. 2023).

First, as RL algorithms are mostly black-box approaches, the loss of a decision maker's control as well as the lack of explainability and trust in the proposed policies can be an issue. Second, RL algorithms require lots of data to be trained including wild exploration, which is a problem in many real-world applications. In this regard, the use of auxiliary environments (cf. digital twins) might be a way to test and to better understand self-learning agents, see Groeneveld et al. (2024). However, the simulated market still needs to be fitted to the actual market which can be a challenge for limited amounts of data.

Third, besides pre-training, even the supervision of agents might be necessary in domains like RM in which decisions can have significant monetary consequences. Moreover, the interaction of self-learning agents is hard to predict making their application somewhat risky. Current research shows that, for instance, effects like algorithmic collusion can occur, cf. Calvano et al. (2020), Kastius and Schlosser (2022), and den Boer et al. (2022). To avoid negative effects of unsupervised training of RL agents, accordingly adjusted reward functions can be used.

Lastly, in specific RM applications, it remains somewhat unclear whether (i) the huge potential of RL algorithms can be effectively used, (ii) the disadvantages, as mentioned above, can be outweighed, and (iii) under which problem setups and data requirements which kind of RL algorithm outperforms well-understood and established forecast-based DP algorithms.

Overall, current issues and challenges of utilizing the potential of RL algorithms in RM applications can be summarized as follows:

- A lack of trust, confidence, and interpretability when applying RL methods.
- A suitably fitted environment for training is needed.
- Amounts of available and required data may not meet.
- Among a variety of available algorithms, the best choice of solution method is not known.
- A partly unclear relation between established classical (forecast-first-then-optimize) DP approaches and state-of-the-art RL algorithms.

The goal of this paper is to address the issues of the last three points by directly compare RL and forecast-based DP methods for exemplary dynamic pricing markets in order to better understand how they relate to each other in terms of (i) required data, (ii) computation time, and (iii) average performance. We seek to investigate in which cases classical DP approaches are still advantageous and—if the problem complexity and the amount of available data is sufficiently large such that RL pays off—which is the best option among a broad variety of algorithm choices.

### Contributions: comparison of classical data-driven dynamic programming and state-of-the-art reinforcement learning methods

In this paper, we compare different data-driven DP and RL methods and how their performances relate to each other depending on the amount of provided data. Our contributions can be summarized as follows.

- We compare the relation of a broad selection of state-of-the-art RL algorithms and data-driven DP methods (based on estimated dynamics) for dynamic pricing applications.
- While we let RL techniques interact with the unknown environment, for the DP methods, we use observational training data to estimate the required model dynamics. This finally allows us to study (i) average rewards, (ii) the amount of required data, and (iii) the computation time

of the different DP and RL methods and compare their overall performance.
- We investigate both monopoly and duopoly market scenarios and discuss the best performing methods from a practitioner's perspective.
- Our numerical evaluation results show that with few data (i.e., in our case less than about 10 episodes of training data), data-driven DP methods remain highly competitive. If more data are available, RL algorithms obtain higher expected rewards, where PPO performs best for up to about 100 episodes. For more episodes of training, DDPG and TD3 provide the best results.
- We provide a flexible open-source simulation and evaluation framework, see code repository https://anonymous.4open.science/r/AirlineSimulation-38CC/.

Considering exemplary problems, our model allows to identify "switching" points regarding the amount of required data, that distinguish cases in which different algorithms are advantageous and thus can be recommended.

This analysis is interesting, especially as firms on the hand still apply established and well-understood DP techniques, while on the other hand, they are interested in exploiting the clear potential of modern RL methods, which however, comes with several problems (data, trust, risk, transparency, etc.), see challenges above.

From a practitioner's view, for example, it is especially interesting which algorithm to use given the competitive scenario and the amount of available training data. From a theoretical point of view, it may also be interesting to observe and to study the critical amount of data for which the results of classical DP methods are outperformed by different RL methods (cf. sample efficiency). In this context, the provided framework can be used to numerically study, for instance, the sensitivity of these critical amounts of data regarding different model parameters, such as booking period, size of the price set (cf. action space), and the number of initial items to sell (cf. state space).

Further, we obtain that about 100 to 1000 episodes are required for the best RL methods to gain more than 90% of the optimal average rewards. Among all RL approaches, PPO is the only candidate that combines fast learning, high rewards, and low computation time.

The remainder of this paper is organized as follows. In Sect. 2, we provide a brief overview of related research topics and considered DP and RL methods. In Sect. 3, we describe our model framework and the underlying Markov decision process (MDP). In Sect. 4, we present experimental evaluations to study the performance of different RL and DP methods in dynamic pricing markets for both monopoly and duopoly setups. In Sect. 5, we summarize the main results, point to limitations, propose future research directions, and discuss model generalizations as well as insights for real-world applications. Finally, Sect. 6 concludes.

## Related work

In the following, we first review the dynamic pricing applications (Sect. 2.1) and then provide an overview of different DP and RL methods to be considered in this work (Sect. 2.2). In Sect. 2.3, we discuss related works that use RL techniques to tackle RM problems. Afterward, in Sect. 2.4, we describe the research gap that we target.

### Dynamic pricing applications

Selling airplane tickets is a classical RM application. An extensive overview on this field and the related field of dynamic pricing research offer the surveys by Chen and Chen (2015), Klein et al. (2020), and Gerpott and Berends (2022). In the academic literature of the airline domain, RM systems have received quite a lot of attention since the deregulation of the airline industry in the 1970 s, see McGill and van Ryzin (1999).

Gallego and van Ryzin (1994) analyze dynamic pricing in a general finite horizon context. They propose an inventory pricing model that delivers nearly optimal policies while considering a stochastic demand. This model explicitly includes airline RM problems. Building up on that, Talluri and van Ryzin (2004) present a choice-based approach by incorporating the consumer behavior into the RM system for solving finite horizon inventory management problems. They further use regression methods like OLS (ordinary least squares regression) to estimate the consumer behavior in monopolistic settings.

Selcuk and Avsar (2019) propose a DP framework for investigating the air travel demand and calculate the optimal policy for four different demand models of monopoly markets. A similar approach pursue Isler and Imhof (2008) while analyzing an airline market with two competing firms. They apply backward induction assuming a full information context to formulate a game theoretical model. For similar markets, Schlosser and Boissier (2018) use a frequent price updating heuristic to compensate for imperfect market anticipations.

### Dynamic programming and reinforcement learning algorithms

**Dynamic Programming (DP) Methods** To address dynamic decision problems with full information, DP methods decompose a multistage problem into a sequence of interconnected one-stage problems. Fundamental for these techniques is the principle of optimality, developed by Richard Bellman, see Bellman (1957). The choice of solution methods for a problem depends on the time horizon and the problem complexity. Finite horizon problems are usually solved by Backward Induction. For the infinite horizon case, typically approaches like, e.g., Policy Iteration (cf. Lagoudakis and Parr 2003) or Value Iteration (cf. Chatterjee and Henzinger 2008) are used.

In case optimal solutions are not tractable anymore, heuristic approaches are applied, e.g., Approximate Dynamic Programming (ADP), cf. Powell (2007). In this context, the book by Powell (2007) provides a comprehensive overview about DP and ADP techniques. In addition, the surveys by Sniedovich and Lew (2006) and Si et al. (2004) deliver further information related to the topics of DP and ADP.

**Reinforcement Learning (RL) Algorithms** In the following, we provide a concise introduction to established RL techniques. In general, in RL, a so-called agent interacts with its environment to achieve a certain goal. The agent undergoes training through a trial-and-error procedure. Therefore, the agent continually engages with an unfamiliar environment, utilizing both observable states and actions, while monitoring feedback signals and the subsequent state transitions. All of this is governed by an underlying Markov process. Simulation-based RL algorithms offer a heuristic approach to address complex Markov decision problems (MDP) with limited information available for the agent.

The book by Sutton and Barto (2018) provides a comprehensive summary of the current state of the art in the field of RL. In this work, we consider the following RL algorithms: Q-Learning (QL), cf. Watkins and Dayan (1992), Deep Q-Learning Networks (DQN), cf. Mnih et al. (2015); Deep Deterministic Policy Gradient (DDPG), cf. Silver et al. (2014), Twin Delayed DDPG (TD3), cf. Fujimoto et al. (2018); Actor Critic (A2C), cf. Mnih et al. (2016); Proximal Policy Optimization (PPO), cf. Schulman et al. (2017), which can be seen as a generalization of A2C Huang et al. (2022); and Soft Actor Critic (SAC), cf. Haarnoja et al. (2018).

## Reinforcement learning in dynamic pricing applications

In real-world markets, full information methods are not directly applicable because the consumer behavior and the competitors' reactions are unknown. Therefore, model-free RM systems gained more attention in the academic literature. The foundations were delivered by Bertsimas and De Boer (2005), proposing a simulation framework to address the stochastic and dynamic character of the consumer demand in the airline RM. Their two key findings are that (i) such simulation-based RM systems are computational feasible and (ii) lead to significant revenue enhancements that opened the door for further simulation-based research. Gosavi et al. (2002) applied Q-Learning (QL) to dynamic pricing problems in the airline domain. They used a semi-Markovian model and showed that QL methods are able to outperform established heuristics in a finite horizon monopoly with inventory considerations.

A comparison of three RL approaches in a dynamic airline pricing market study is presented by Collins and Thomas (2012). Concluding that Q-Learning and State-Action-Reward-State-Action (SARSA) techniques outperform Monte Carlo Learning in their market setup, they also mention that traditional methods become unsuitable for solving more complex models. Collins and Thomas (2013) apply SARSA on a complex game including multiple varying customer aspects, warning of extra dimensions impacting the unpredictable ability of solving such a game with the studied algorithm.

A more extensive application of RL algorithms on airline markets offer Bondoux et al. 2020. With their proposed simulation framework, they apply QL and Deep Q-Learning Networks (DQN) algorithms and compare them against an optimal benchmark in monopoly and duopoly settings. To obtain the optimal policy, they use DP techniques. Bondeaux et al. further use DP to initialize the neural network of the DQN algorithm to minimize the exhaustive training efforts which are needed when applying RL algorithms in RM systems. A limitation of their work is the narrow selection of RL algorithms as well as the RL environment. In the state they include the time and number of bookings which does not map the Markov process fully to the RL environment.

Dynamic pricing in the airline domain is further studied by Shihab and Wei (2022) using DQN or Lawhead and Gosavi (2019) using advanced actor critic methods. Their results for monopoly markets show that classical expected marginal seat revenue-b (EMSRb) heuristics can be outperformed. Approaches to further scale DQN approaches are proposed by Nataraj et al. (2024) using transfer learning techniques. Extensions to address multi-flight scenarios under the Multinomial Logit (MNL) choice model are investigated by Zhu et al. (2024).

Another approach using Deep Q-networks (DQN) and SARSA techniques to learn pricing policies and ordering quantities (in a monopoly market with strategic consumers) was done by Famil Alamdar and Seifi (2024), including the modeling of substitute products. Their calculation with markdown prices for used products in the second of two combined decision periods and the inclusion of inventory considerations show that DQN and SARSA are suitable for solving large-scale pricing optimizations. Further, the works by Zhou et al. (2022) and (2023) use PPO and Double Deep Q-Networks-based techniques to study joint dynamic pricing and inventory control problems with reference price effects.

Strategic behavior on the consumer side is also studied by Kong et al. (2020), utilizing long short-term memory networks (LSTMs) to predict a customer's response in a RL market simulation framework. In their conclusion, they highlight that the blind exploration of RL is not sufficient to learn daily changes in their practical model of load reduction. Proposing an approach to estimate customers' response with LSTMs, they virtually explore optimal prices resulting in less optimization for myopia. What is more, in the recent work by Yavuz and Kaya (2024), the authors use RL for pricing and inventory control of perishable products, investigating Q-Learning and SAC. Comparing the learning policies to an optimal solution obtained by DP methods, they conclude that SAC performs faster and achieves near-optimal solutions in their experiments.

### Research gap

Overall, an increasing number of articles study the application of RL algorithms in the RM and pricing domain. While the potential of RL has clearly been shown remaining challenges, see above, explain that established DP methods are still widely used in practice. In this context, a comparison between DP and RL methods would help practitioners to make more informed decisions regarding the choice of algorithms available.

Although studies comparing RL algorithms exist [e.g., Zhu et al. (2024) for DQN, PPO, A2C, and TRPO or Groeneveld et al. (2024) for SAC, A2C, and PPO], to the best of our knowledge, it is hardly studied how the results of various RL techniques compare to classical DP approaches—which estimate first and then optimize—with regard to expected rewards and required data. For this reason, it is unclear which methods perform best in certain settings, e.g., under competition, and how much training data are required. Our goal is to address this research gap using a broad comparison of different methods in a reproducible basic market setup.

## Model and problem description

We model a market for selling airplane tickets for flights with equivalent (or comparable) destinations. Within a booking period, airlines (players) set prices for tickets on the marketplace. Arriving customers decide whether to buy a ticket for the offered prices or not. This basic model allows to describe market scenarios in monopoly and duopoly settings.

Active decisions are made by the firms and the consumers. For determining the pricing decisions, we use different DP and RL solution methods. We then further analyze and evaluate the resulting pricing strategies for different solution methods. In duopoly settings, we consider rule-based strategies for the competitor. The consumer behavior models heterogeneous customer groups with a varying willingness-to-pay. We consider customers of a myopic type with price- and time-based decisions.

Next, in Sect. 3.1, we describe our model and the consumer behavior in detail. In Sect. 3.2, we describe the application of DP and RL solution within the considered usecase.

### Model description

In the following, we describe the setup of our marketplace and formulate the decision problem as a Markov Decision Process (MDP). It consists of formulations of states, actions, rewards, and state transitions, cf., e.g., Sutton and Barto (2018). Our dynamic pricing model follows classical standard frameworks and assumptions—such as discrete time, finite horizon, monopoly/duopoly setups, time-dependent demand, MNL-based customer choice models, and myopic customers—see, e.g., Gallego and van Ryzin (1994), Klein et al. (2020), and Talluri and van Ryzin (2004).

#### Marketplace setup

We consider a finite time horizon $T$ which represents the booking period in the airline market. E.g., in many applications a period refers to days. The discount factor for one unit of time is 1. The modeled marketplace is further described by the parameters in Table 1.

**Table 1** Parameters with brief description and default values of our market model

| Symbol | Description | Default value |
|---|---|---|
| $T$ | Length of the time horizon (booking period in, e.g., days) | 20 |
| $s^{max}$ | Inventory level (seating capacity) of an airline in $t = 0$ | 20 |
| $i_{con}$ | Number of arriving consumers per period (time step) | 5 |
| $p^{max}$ | Maximum price for airline tickets | 20 |
| $p^{grid}$ | Grid size for equidistant price sets $A_{dis}$ | 1 |

### Monopoly

The monopoly is the basic market scenario in our simulation. Here, in interaction with consumers, the monopolistic airline needs to find a pricing policy that maximizes long-term sales revenues. Note that our simulation framework allows to examine market settings where the airline has full knowledge about the marketplace as well as settings where demand and market parameters are unknown.

**State** $s_t$ In the beginning of the booking period at time step $t = 0$, the airline can offer all seats for the flight (i.e., $s^{max}$ denotes the initial inventory level). With customers purchasing tickets, the number of available seats decreases. In the MDP framework, the seating capacity over time is described by the set of states $S_{mon}$. As in reality the seating capacity is limited, $S_{mon}$ is modeled finite and discrete. With the seat capacity decreasing over time, we consider the environment's state, $t = 0, 1, ..., T$, $s_t \in S_{mon} := \{0, 1, 2, ..., s^{max}\}$.

**Action** $a_t$ Depending on the pricing policy, the airline varies prices accordingly. Without loss of generality, we assume that an airline can change its price once per time step. In the MDP framework, the price range is represented by the set of actions $A$. Our simulation framework allows $A$ to be modeled discrete, containing a finite set of prices, or continuous, as a continuous interval. This allows to apply various DP and RL algorithms. In the discrete case, the price granularity—i.e., the distance between two adjacent prices—is defined by $p^{grid}$, i.e., $A = A_{dis} := \{p^{grid}, 2 \cdot p^{grid}, ..., p^{max}\}$. The maximum price $p^{max}$ should be sufficiently large such that the consideration of higher prices will not significantly influence a model's solution, cf. null price. In the continuous case, we use the set of prices

$$A = A_{con} := (0, p^{max}]. \tag{1}$$

**Reward** $r(i_t, a_t, s_t)$ A player's revenue is determined by the amount of tickets they can sell to which price. Transferring that to the MDP framework, the reward $r(i_t, a_t, s_t)$ per time step $t$ results in the number of sold tickets and the selected action. The number of sales is constrained by (i) the number of arriving customers per period—in our model denoted by $i_{con}$—as well as (ii) customers' buying preferences. The consumer buying preference $d(a_t, t)$ depends on the price and the time and includes a no-buy option. The consumer behavior and the choice of $d(a_t, t)$ are further described in Sect. 3.1.5.

We express the number of customers within a period willing to buy a ticket $i_{buy}$ (or not willing to buy $i_{no\ buy}$) as a tuple $i := (i_{buy}, i_{no\ buy})$ of the event space $I^2$. We model possible events as a discrete, finite set $I$ constrained by the number of arriving consumers $i_{con}$, i.e., $i \in I_{mon} := \{i = (i_{buy}, i_{no\ buy}) \in I^2 | i_{buy} + i_{no\ buy} = i_{con}\}$, with $I := \{0, 1, 2, ..., i_{con}\}$. Building on that, the reward per time step $t$ is defined as

$$r(i, a_t, s_t) = a_t \cdot min(i_{buy}, s_t). \tag{2}$$

**State transition** $s_{t+1}$ With customers purchasing tickets in time step $t$, the airline can only offer remaining seats in the next time step $t + 1$. Therefore, the state variable, cf. $s_t$, decreases by number of sold tickets $min(s_t, i_{buy})$ when transitioning to the next state $s_{t+1}$, i.e.,

$$s_{t+1} = s_t - min(s_t, i_{buy}). \tag{3}$$

Recall, DP methods use the information about state, reward probabilities, and state transition probabilities. Instead, RL methods, in each step, get information about the current state as well as the realized reward signal and the follow-up state associated to the chosen action.

### Duopoly

We model a market in which two airlines offer prices for a comparable one-fare flight and therefore compete against each other. Hence, each airline seeks to find prices that maximize its expected long-term reward in the presence of competition. Moreover, as in the monopoly variant, the booking period ends when the final time step $t = T$ is reached, regardless how many tickets an airline has sold. If $i_{con}$ and $T$ are comparably small, competitive pricing strategies that undercut the competitor's prices are key. The associated MDP is an extended version of the monopoly model, with the following changes.

**State** $s_t$ For the duopoly, we extend the state space to include the seating capacities $s_t^{(a)}$ for firm 1 (i.e., the (a)gent) and $s_t^{(c)}$ for firm 2 (i.e., the (c)ompetitor), respectively. The state also includes the *last* action of the competitor $a_{t-1}^{(c)}$ for period $(t-1, t)$. We assume that for each period both players select actions from the action space introduced for the monopoly case $a_t^{(a)}, a_t^{(c)} \in A$. Hence, we consider the environment's state space

$$S_{duo} := \left\{ s_t = \left(s_t^{(a)}, s_t^{(c)}, a_{t-1}^{(c)}\right) \in S_{mon} \times S_{mon} \times A \right\}.$$

Note, both firms base their price decision for the actual period on the own inventory and the opponent's last price; they do not know the other firm's new price. In the duopoly case, depending on realized sales, the environment's **state transition** $s_{t+1}$ becomes

$$s_{t+1} = \left(s_t^{(a)} - min(s_t^{(a)}, i_t^{(a)}), s_t^{(c)} - min(s_t^{(c)}, i_t^{(c)}), a_t^{(c)}\right). \tag{4}$$

**Reward** $r(i_t, a_t, s_t)$ We apply the same reward function, cf. (2), to both players. However, in the presence of a competitor, the event space $I$ needs to be adjusted. Each arriving customer can now choose between the

**Table 2** Rule-based competitors that are implemented in our simulation framework

| Name | Price reaction $a_t^{(c)}$ | Description |
|---|---|---|
| Fixed price | $a_{const}$ | Constant |
| Undercut | $max(a_{thresh}, a_t^{(a)} - p^{grid})$ | Price-dependent |
| Storage | $if \quad s_t^{(c)} > 0 \quad then \quad \frac{p^{max}}{p^{grid}} \cdot p^{max} \cdot \frac{1}{4} \quad else\ p^{max}$ | Storage-dependent |
| Early undercut | $if \quad t < \frac{T}{2} \quad then \quad max(a_{thresh}, a_t^{(a)} - p^{grid}) \quad else \quad p^{max}$ | Price- and time-dependent |
| Advanced undercut | $if \quad \theta < 1 \quad then \quad max(p^{max} \cdot \frac{1}{3}, a_t^{(a)} - p^{grid}) \quad else \quad max(p^{max} \cdot \frac{1}{6}, a_t^{(a)} - p^{grid})$ | Price- and market-dependent |

Parameter $\theta$ measures whether time $T$ is small (or large) compared to the inventory $s^{max}$—then the policy is more aggressive; we use $\theta = \frac{T}{s^{max}}$

**Table 3** Exemplary demand groups of our framework; the corresponding parameters are defined as follows: $\alpha_a = 1 - \frac{a}{p^{max}}$, $\beta_t = \frac{3}{2} - \frac{3t^2}{2T} - \frac{3}{4}T, \gamma_t = \frac{1}{4} \cdot t - \frac{T}{4}, \delta_a = \frac{1}{a+1}$, and $\epsilon = \frac{1}{\ln(T+1)}$

| Index $i$ | Name | Demand model $d_i(a,t)$ | Description ($i = 1, ..., 5$) |
|---|---|---|---|
| 1 | Standard | $\alpha_a \cdot (1+t) \cdot \frac{1}{T}$ | Price- and time-based |
| 2 | Rational | $\alpha_a$ | Price-based |
| 3 | Family | $\alpha_a \cdot \exp(\beta_t)$ | Price- and time-based |
| 4 | Business | $\exp(\gamma_t)$ | Time-based |
| 5 | Early booking | $\delta_a \cdot \ln(t+1) \cdot \epsilon$ | Price- and time-based |

no-buy option, cf. $i_t^{(0)}$, purchasing a ticket from the airline, cf. $i_t^{(a)}$, or purchasing from the competitor, cf. $i_t^{(c)}$. Thus, we have a 3-dimensional event space $I_{duo}$, i.e., $I_{duo} := \{i_t = (i_t^{(0)}, i_t^{(a)}, i_t^{(c)}) \in I^3 | i_t^{(0)} + i_t^{(a)} + i_t^{(c)} = i_{con}\}$, with a well-defined probability distribution that depends on time as well as both competitor's current prices.

## Competitor's strategy

Our framework supports rule-based strategies which depend on time $t$, the opponents action $a_t^{(a)}$, and the his/her inventory $s_t^{(c)}$. Table 2 shows the current implemented strategies which can be applied as competitor strategies in duopoly settings. Note, these can be easily extended.

## Consumer behavior

We consider a stream of $i_{con}$ consumers arriving at each time step $t$. The consumer behavior can be defined as deterministic or stochastic. For both cases, we use probabilities that are defined below, depending on the demand model. The buying decisions of all customers are distributed proportionally to those probabilities in the deterministic case and drawn multinomially in the stochastic one. A single customer arriving at time $t$ observes the current offered prices $\mathbf{a} := (a_t^{(a)}, a_t^{(c)})$ from both firms. A consumer's choice can be arbitrarily defined and does include a no-buy option. Our simulation framework includes predefined exemplary non-trivial demand models, which are shown in Table 3.

The overall consumer behavior is expressed as a probability distribution for each single arriving consumer to buy no ticket $P_{no\ buy}$ or to buy a ticket from one of the two airlines $P_{buy}^{(k)}$, $k = 1, 2$. Given some current prices $\mathbf{a}$ of the firms, the probabilities are defined such that

$$P_{no\ buy}(\mathbf{a}, t) + P_{buy}^{(1)}\left(\overline{d}(a_t^{(a)}, t)\right) + P_{buy}^{(2)}\left(\overline{d}(a_t^{(c)}, t)\right) = 1. \qquad (5)$$

All demand groups, cf. Table 3, can be combined or weighted in an arbitrary fashion, such that for each $D \subseteq \{1, 2, 3, 4, 5\}$:

$$\overline{d}(a, t; D) = 1/|D| \cdot \sum\nolimits_{i \in D} d_i(a, t). \qquad (6)$$

Note, the simulation framework can be easily extended with further consumer choice models.

## Objective

A single airline's rewards are characterized by the sales revenues it gains over the booking period $t = 0, ..., T$. Given a pricing policy $\pi = \pi_t(s_t)$ to choose $a_t$ the future rewards, cf. (2), are

$$G_t^{\pi} = \sum\nolimits_{k=0,...,T} r(i_k, a_k, s_k). \qquad (7)$$

The objective of an airline is to find a pricing strategy that enhances a provided goal, such as maximizing the expected total profits from the initial state $s_0$ on

$$E(G_0^{\pi} | s_0). \qquad (8)$$

The initial state at the beginning of the booking period $t = 0$ is determined by the inventory levels $s^{max}$ of both airlines and the first action $a_0^{(c)}$ according to the competitor's policy, $s_0 := (s^{max}, s^{max}, a_0^{(c)})$. The initial state in a monopoly skips the entries 2-3 of the tuple. Note, an optimal policy can usually only be guaranteed if a player has full knowledge about the market dynamics. This includes full information about the marketplace, the consumer behavior, and competition if applicable. DP-based methods, e.g., backward induction, deliver optimal solutions in these contexts. In scenarios without full information (usually consumer behavior and competitor reactions are unknown), RL algorithms can be used as heuristics to solve the problem. Furthermore, we use demand estimations (e.g., via basic linear regressions) to be able to apply DP algorithms to refrain from full information contexts.

## Application of solution methods

Based on the model description and problem formulation in Sect. 3.1, we describe how different solution methods can be applied to our problem.

### Dynamic programming (DP)

DP methods are well explored in the pricing context, cf. Gallego and van Ryzin (1994) or Isler and Imhof (2008). We use tabular DP methods (i.e., backward induction) in our model framework to obtain the pricing policy, that optimizes the expected total rewards, cf. (8). Having an optimal policy we can compare different heuristic methods that work in none-full information settings to evaluate how well they solve the problem. Additionally, as an approximate dynamic programming (ADP) method, we test forward dynamic programming, see, e.g., Powell (2007). Here, we use a choosable number of simulations of the sales horizon of $T$ periods (cf. episodes) to approximate the value function.

To deal with an unknown environment, we estimate the model's dynamics from data. Here, standard estimation methods like OLS are explored in the context of airline RM, see, e.g., Talluri and van Ryzin (2004). Particularly, the estimation of the consumer behavior based on historical data samples is essential to the application of DP techniques without full knowledge of the MDP (the evolution of the inventory, cf. #new items =#old items—#sales, is trivial). This enables us to compare DP-based methods and RL methods in a fair way.

### Reinforcement learning (RL)

RL algorithms are applicable in the contexts of dynamic pricing, cf. Kastius and Schlosser (2022), and airline RM, cf. Bondoux et al. (2020). The described discrete time MDP (Sect. 3.1) is suitable for standard RL frameworks, such as Stable Baselines, cf. Raffin et al. (2021). Since the MDP is time discrete and RL frameworks require a turn-based environment, we simulate the booking period repeatedly in turns, so-called episodes. With the gym environment (Brockman et al. 2016), our implementation framework provides a standardized API for applying various state-of-the-art RL algorithms. Our evaluations are shown in the next section.

## Evaluation

In this section, we show our experimental result to showcase the comparison of DP and RL methods for dynamic pricing problems in exemplary monopoly and duopoly scenarios. The training setup and all considered algorithms are introduced in Sect. 4.1. In Sect. 4.2, we show demand estimation results, which are used to fit the market dynamics to be used in DP methods without full initial knowledge. Then in Experiment A, we present performance comparisons for a monopoly setup (Sect. 4.3). Further, Experiment B evaluates a duopoly setup (Sect. 4.4).

## Selected methods and training

The baseline for our experiments—the optimal pricing strategy—is obtained by applying a backward induction. Proceeding from this baseline, we are able to compare DP and RL techniques, see Table 4. The algorithms of type RL* require a continuous action space, cf. (1). For the evaluation of those methods, the actions are rounded in order to be comparable to the other methods. We use the default hyperparameters of the algorithms for our experiments. They are summarized in Tables 8, 9, 10, 11, 12, 13, and 14, which are given in the Appendix. We train the RL algorithms as well as the forward DP approaches for up to 1000 episodes with $T$ steps per episode.

Equivalently, we use up to $1000 \cdot T$ data points for the estimations for DP methods. To be able to evaluate the performance of each method given their resource consumption (in terms of consumed data), we analyze the training process after $T, 10 \cdot T, 100 \cdot T$, and $1000 \cdot T$ steps, respectively, data points. We run each training $n = 5$ times and average the results.

**Table 4** Overview of the different DP and RL methods used in our experiments (*denotes a continuous price set; otherwise a discrete price set is used)

| Type | Name | Method |
|---|---|---|
| DP | DPopt | Backward induction with true demand |
| DP | DPest | Backward induction with estimated demand |
| DP | ADP | ADP (forward DP) with true demand |
| DP | ADPest | ADP (forward DP) with estimated demand |
| RL | QL | Q-learning (cf. Watkins and Dayan 1992) |
| RL | DQN | Deep Q-learning (cf. Mnih et al. 2015) |
| RL* | DDPQ | Deep deterministic policy gradient (cf. Silver et al. 2014) |
| RL* | TD3 | Twin delayed DDPG (cf. Fujimoto et al. 2018) |
| RL* | A2C | Actor critic (cf. Mnih et al. 2016) |
| RL* | PPO | Proximal policy optimization (cf. Schulman et al. 2017) |
| RL* | SAC | Soft actor critic (cf. Haarnoja et al. 2018) |

### Demand estimation and fitted environments

To fit the model dynamics from data, we aim for estimating the probabilities $P_{no\ buy}$ and $P_{buy}$ in the monopoly case to subsequently use it in our methods. To do that, we proceed in two steps: Firstly, we estimate the average number of customers buying out of the total number of customers using basic OLS. In the second step, we transform this estimation of the mean into probabilities using a Poisson distribution.

The regression uses a firm's data samples (observed own sales) in period $t$ under action $a$ to predict the event $i$. In the monopoly case, those are the values for the current timestep and the price offered by the firm. In the duopoly case, the offer price of the competitor is additionally taken into account. We here assume that the current offer price of the competitor is observable (in practice ticket prices are public). Alternatively, a competitor's price reactions could be predicted using historical data. Further, we decided not to predict the demand explicitly for the competitor offer. Assuming a fair setup, the estimated demand can also be used for the competitor. Finally, the inventory of the competitor is, in general, not observable. Note, to obtain an optimal policy as upper bound, this information could be made observable.

In the following, we describe how our exemplary demand estimation is organized. Each of the available input data variables (i.e., price and time) is transformed by basis functions to a subset of features (cf. linear basis functions), which add up to the total feature set when combined. Those transformations for one data variable $z$ include $z^2$, $\sqrt{z+1}$, $\log(z+1)$, and several combinations of multiplications of the variables, e.g., $a \cdot t$ in the monopoly case with price $a$ and time $t$. In the duopoly case, the number of features used is 22 and includes both firms' prices (for more details see our code repository). To produce training data with sales realizations at different periods $t$, we used uniformly distributed prices.

The goal of the regression is to find a coefficient for every of those features and an intercept to estimate the number of buying customers, i.e., sales events $i$ accurately. Therefore, we explored the performance of OLS, Ridge, and Lasso regressions, which have varying degrees of regularization with respect to the Mean Squared Error (MSE) and the $R^2$ -Score (R2).

In the second step, we turn this estimation of average sales into probabilities for all possible events $i$. We use a Poisson distribution with estimated mean $\hat{\lambda}_t$, i.e., we have, $i \geq 0, a \in A$,

$$P_t(i,a) = \frac{\hat{\lambda}_t(a)^i \cdot e^{-\hat{\lambda}_t(a)}}{i!}.$$

Finally, the estimated sales probabilities $P_t(i,a)$ can be used within a fitted model without knowing the true demand.

**Table 5** Average normalized MSE values (bold) and standard deviations ($\sigma$) for 100 repeated regressions fitted on $N$ random datapoints (periods) of a monopoly model

| $N$ | OLS | Ridge | Lasso |
|---|---|---|---|
| 10 | **178.57** $\sigma$ 618.5 | **0.909** $\sigma$ 2.229 | **0.542** $\sigma$ 0.752 |
| 100 | **0.3025** $\sigma$ 0.2 | **0.249** $\sigma$ 0.098 | **0.320** $\sigma$ 0.105 |
| 1000 | **0.2119** $\sigma$ 0.023 | **0.215** $\sigma$ 0.023 | **0.318** $\sigma$ 0.033 |
| 5000 | **0.2084** $\sigma$ 0.009 | **0.209** $\sigma$ 0.008 | **0.309** $\sigma$ 0.015 |
| 10000 | **0.2082** $\sigma$ 0.007 | **0.210** $\sigma$ 0.007 | **0.312** $\sigma$ 0.009 |

Lasso and Ridge refer to the standard L1 and L2 regularization variants of OLS. All regressions required less than one second computation time

In duopoly scenarios, the demand prediction for $P_t^{(k)}(i, \mathbf{a})$, $k = 1, 2$, is done similarly. If a firm has an inventory of 0, we can set a high price (cf. null price), e.g., $p^{max}$, or switch to monopoly demand estimations for the remaining firm.

### Experiment A: Comparison results for a monopoly market

In this experiment, we examine a monopoly market with moderate problem, i.e., $|S_{mon}| = 21$, $|A_{dis}| = 20$, $T = 20$, and $i_{con} = 5$ arrivals per period.[1] On this monopoly market, we apply the following solution methods: (i) DP with and without estimation, (ii) ADP with and without estimation, and (iii) state-of-the-art RL algorithms. Accordingly to that, all methods from Table 4 are applied in this experiment. The goal of this experiment is to select the best performing approaches to apply them in a market with competition in Experiment B.

Next, we first describe the considered demand setup (Sect. 4.3.1) and show the associated regression results (Sect. 4.3.2), which are used by the data-driven DP methods. Finally, we present our evaluations (Sect. 4.3.3) and summarize the results obtained (Sect. 4.3.4).

#### Demand settings

We use the configuration of the marketplace presented in Sect. 3.1.1, cf. Table 1. The consumer behavior is defined as follows: We use the demand model introduced in Sect. 3.1.5 with a single firm. We consider $i_{con} = 5$ customer arrivals per time step $t$ with an independent buying behavior sampled from the probabilities $P_{no\ buy}$ and $P_{buy}$, cf. (5). To obtain these probabilities, we use the utility (or score) function $\overline{d}(a,t)$, cf. (6), with $|D| = 5$ in combination with the softmax

[1] Algorithms of type RL* (see Table 4) require a continuous action space, therefore is $A = (0, p_{max}]$, cf. (1).

function. We use $P_{no\ buy} = e^1/\sum$ and $P_{buy} = e^{\bar{d}(a,t)}/\sum$, where $\sum := e^1 + e^{\bar{d}(a,t)}$.

## Demand estimation results

In the monopoly case, we found that OLS and Ridge regression perform equally and better in comparison to Lasso (cf. Table 5). The MSE becomes sufficiently small with at least 100 data points. We decided to continue with OLS because of its prevalence in regression use cases. A regression fitted with OLS on 1000 datapoints reaches an R2 value of 0.837 without stochastic customers and 0.35 with stochastic customers. MSE values tend to be larger in the duopoly case and more data points are necessary to produce an accurate estimation. With 5000 data points of a stochastic environment, we achieve a normalized MSE of 0.177 with a standard deviation of 0.00975 and an R2 value of 0.2663 with a standard deviation of 0.0335.

## Performance analysis

In this section, we evaluate and compare the results of the different DP and RL algorithms for the considered monopoly setup. Here, we focus on (i) average rewards, (ii) computation time, and (iii) the impact of the number of training episodes.

Average rewards are evaluated as follows. After the training of a method, where we use $x \in \{1, 10, 10^2, 10^3\}$ episodes, we evaluate the obtained pricing strategy by simulating 100 episodes with $T = 20$ (computation time of 20 s) in our market with stochastic demand. The mean of the 100 rewards obtained within these evaluation runs then represents the *simulated reward* of the method. This way, we also evaluate standard deviations of rewards.

Figure 1 shows the simulated reward of each method's pricing strategy gained in our simulation (y-axis) after a specified amount of training episodes (x-axis). The black vertical lines on top of each bin show the standard deviation per value. The expected reward of 250 gained by the optimal solution (benchmark) is 250.

In the beginning of the training, the performances of the DP methods and the RL algorithms QL, DQN, and A2C are characterized by a high deviation of the performance within the 5 training runs. This is due to the little amount of 20 data points that each method has observed in the first episode. Here, the highest reward of 184 is reached by PPO. This can be explained by the high sample efficiency of the PPO algorithm and the fact that DPest suffers from a badly fitted market environment. Nevertheless, results for the extreme case of one episode only should be handled with care.

For 10 training episodes, DPest is the best performing approach in this interval of available data followed by PPO both reaching rewards over 170. The performance increase of DPest is caused by a better fit of the model based on the increased number of observations, cf. the strong decrease of the MSE in Table 5 from 10 to 100 data points. What further stands out is the dip in the rewards of the RL methods DDPG, SAC, and PPO. This can be explained by their off-policy design which brings faster learning speed as well as instability in training.

With proceeding training progress (i.e., with more than 100 episodes), more methods reach that reward over 170 which represents ≈ 67% of the optimal benchmark. While the quality of the model fit of DPest does not further increase with more data, cf. also Table 5, the additional data allow the

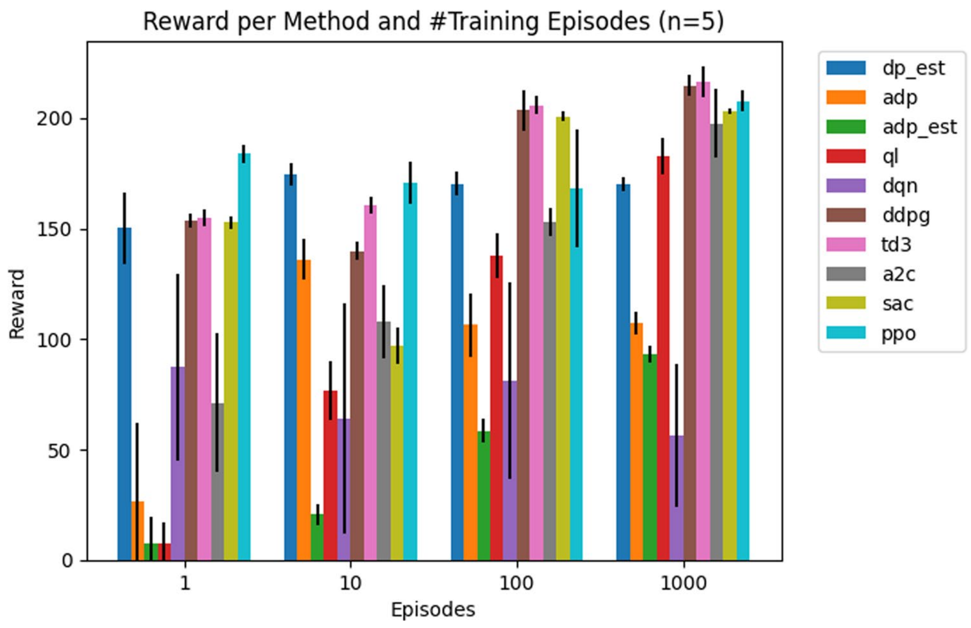

**Fig. 1** Monopoly Case: Average rewards (for 100 simulated episodes) of pricing strategies obtained by different DP and RL methods for different lengths of training time (1, 10, 100, and 1000 episodes). The benchmark DPopt with full information gains an optimal expected reward of 250. Results were averaged over $n = 5$ separate training runs and their evaluations

RL algorithms, i.e., PPO, TD3, SAC, and DDPG to achieve the highest mean rewards from $\geq 100$ training episodes on. Here, even the performance of QL surpasses the results of DPest. Finally, the best reward of 216 achieved by TD3 reaches 86% of the optimal benchmark after 1000 episodes of training.

The performance peak of DPest from 10 episodes on can be explained by the error rates of the linear regression. As shown in Table 5, the MSE values significantly decrease with data from 10 episodes to 100 episodes. In comparison to that, the decrease of the MSE for more than 1000 episodes seems to be insignificant. Accordingly, the rewards of DPest do not further increase with more data. In our examples, we further observe that ADP and ADPest do not provide strong results.

Looking at training times, we find that PPO, A2C, DQN, and QL were the fastest RL approaches with less than 20 s for 1000 episodes of training. Compared to that, TD3, DDPG, and SAC required way more computational time in our setting, see Table 6, with less than 200 s for 1000 episodes of training. For the considered problem size, the solution time for DPopt and DPest was—with 17 s in total—comparably fast. The other simulation-based DP methods, i.e., ADP and ADPest, required 154 and 171 s, respectively, for 1000 episodes.

### Summary of results (monopoly case)

A summary of our evaluation results for the monopoly experiment is given in Table 6. To aggregate the outcomes over all examined training setups, i.e., $\{1, 10, 10^2, 10^3\}$ training episodes, we averaged the respective values for simulated rewards gained by each method (cf. $\bar{r}_{sim}$). The methods using estimations are fitted, respectively, on the amount of data points that $\{1, 10, 10^2, 10^3\}$ training episodes generate, in our case with $T = 20$, resulting in $\{20, 200, 2 \cdot 10^3, 2 \cdot 10^4\}$ data points. Half of the methods were able to achieve results where $\bar{r}_{sim} \geq 150$. In this aggregated comparison, TD3 and PPO achieved the best results regarding rewards and outperformed DPest by about 10%.

Besides the performance of the methods, Table 6 also provides information about the time consumption and, hence, the scalability of the methods. We recall that tabular DP algorithms badly scale with the problem size, cf. curse of dimensionality, i.e., due to high computation times they are not applicable to larger problems. In comparison, the applicability of RL algorithms is less affected by the problem size. Instead, the required computation time highly depends on the training times. In this regard, QL, DQN, A2C, and PPO allow much longer training runs compared to TD3, DDPG, and SAC, which makes them more scalable.

Finally, the considered ADP methods remain due to their simulation-based character applicable to larger problems; however, their learning is much less effective compared to the RL methods and the raining is also comparable slow. Hence, in our comparison, ADP methods can be ruled out. Further, QL, DQN, as well as A2C did not provide promising results. Among the best performing RL approaches, PPO is the only candidate that combines fast learning, high rewards, and low computation time.

## Experiment B: Comparison results for a duopoly market

In this experiment, we compare the best performing methods from Experiment A—i.e., DPest, DDPG, PPO, SAC, and TD3—and evaluate them in a duopoly setting against the same rule-based competitor. Furthermore, we include

**Table 6** Performance summary (monopoly case): we equally weight the average reward results of Fig. 1 (after 1, 10, 100, and 1000 training episodes) for each method obtained within Experiment A using the marketplace setup defined in Table 1

| Name | Mean simulated rewards ($\bar{r}_{sim}$) | Environment | Training time in s (for 1000 episodes) | Scalability |
|---|---|---|---|---|
| DPopt | 250.0 | Original (known)* | 17 (total) | (No |
| TD3 | 184.4 | Original | 210 | (Yes |
| PPO | 182.5 | Original | 20 | (Yes) |
| DDPG | 177.8 | Original | 207 | (Yes) |
| SAC | 163.4 | Original | 320 | (Yes) |
| DPest | 166.2 | Fitted | 17 (total) | (No |
| A2C | 132.3 | Original | 24 | (Yes |
| QL | 101.1 | Original | 001 | (Yes |
| ADP | 93.9 | Original (known)* | 154 | (Yes) |
| DQN | 72.3 | Original | 001 | (Yes |
| ADPest | 45.0 | Fitted | 171 | (Yes) |

The training time is given in seconds per 1000 episodes for RL methods; for DP methods it is the total time. *While for DPopt and ADP, the original environment is known, for all other methods it is not known

ADPest to test the performance of our estimation. To be able to train and to evaluate all approaches multiple times in a feasible time (due to limits of computational power and tractable optimal benchmark solutions), we consider a smaller marketplace for this experiment.

## Demand settings

For the duopoly setup, we use the market configuration shown in Table 7. The consumer demand is modeled as in Sect. 4.3.1 described in detail. In the duopoly case, we have probabilities for the no-buy option $P_{no\ buy}$ and for the cases that either the airline sells $P^{(1)}_{buy}$ or that the competitor sells $P^{(2)}_{buy}$. Using the utility function $\overline{d}(a,t)$, cf. (6), we define the probabilities following standard multinomial logit (MNL) models. Using $\sum := e + e^{\overline{d}(a_t^{(a)},t)} + e^{\overline{d}(a_t^{(c)},t)}$ with $a_t^{(a)}$ the action of the airline (agent) and $a_t^{(c)}$ the competitor's action, we obtain the probabilities $P_{no\ buy} = e/\sum$, $P^{(1)}_{buy} = e^{\overline{d}(a_t^{(a)},t)}/\sum$, and $P^{(2)}_{buy} = e^{\overline{d}(a_t^{(c)},t)}/\sum$. As a competitor strategy, we use the rule-based strategy "Advanced Undercut" as defined in Table 2.

## Performance comparison (duopoly case)

Figure 2 shows the simulated rewards of pricing policies for different methods (and different training stages). There metrics are described in detail in Sect. 4.3.3. As before in Experiment A, the early training stages are characterized by the dominance of PPO which can be seen as "robust" in situations with scarce training data. From 10 training episodes (respectively 100 data points) on, the estimation of sales probabilities is good enough to enable DPest to achieve rewards similar to PPO. In later stages of the training (100 and more episodes), off-policy RL algorithms like TD3 and DDPG outperform both DPest and PPO. The training and evaluation times are slightly higher than in Experiment A; the relations of the computation times, however, remain similar to each other.

Note, although $T$, $A$, and $s^{max}$ are smaller than in Experiment A, the problem is not necessarily easier due to the competitive and, hence, more complex character of the market caused by the presence of a (rule-based) competitor. Note, in this finite horizon duopoly market, the challenge is balance (i) undercutting competitor's prices to sell items and (ii) induce and exploit a runout of the competitor to further act as

**Table 7** Parameters configuration of the duopoly experiment in a basic test market

| Symbol | Explanation | Default value |
|---|---|---|
| $T$ | Length of the time horizon (booking period) | 5 |
| $s^{max}$ | Inventory level (seating capacity) of an airline in $t = 0$ | 5 |
| $i_{con}$ | Number of arriving customers per time unit | 5 |
| $p^{max}$ | Maximum price for airline tickets | 5 |
| $p^{grid}$ | Grid size for equidistant price sets $A_{dis}$ | 1 |

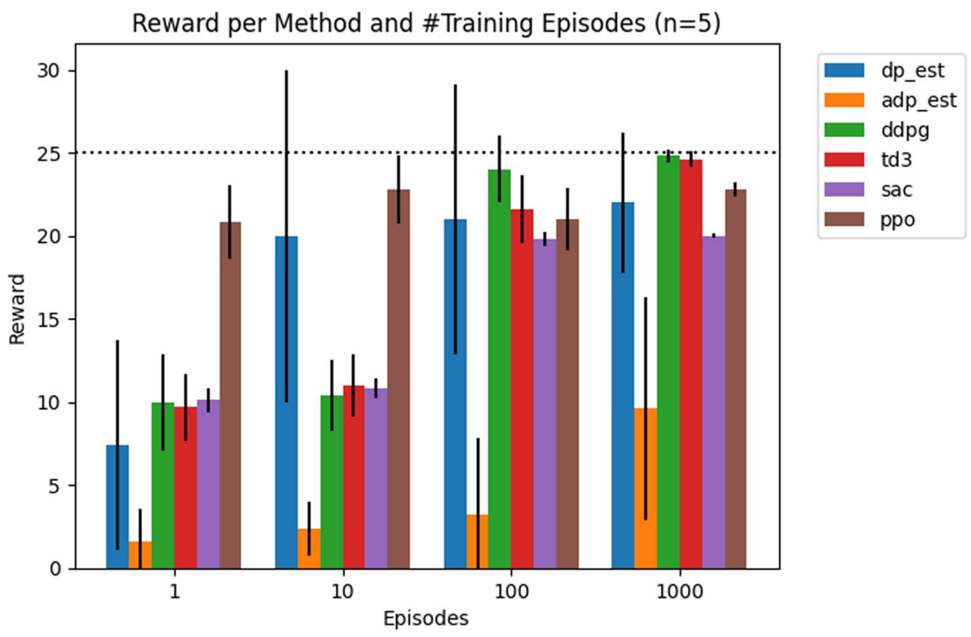

**Fig. 2** Duopoly Case Performance: Average rewards (for 100 simulated episodes) of the pricing strategies obtained by different DP and RL methods for different lengths of training time (1, 10, 100, and 1000 episodes). DPopt with full information gains an optimal expected reward of 25. Results were averaged over $n = 5$ separate training and evaluation runs

a monopolist for the remaining time, cf. Schlosser and Richly (2019). Overall, we obtain that—similar to Experiment A—about 1000 episodes are required for the best RL methods to obtain more than 90% of the optimal average rewards.

## Discussion

Next, we highlight our main insights (Sect. 5.1), discuss limitations of the current model (Sect. 5.2), and provide an overview about possible extensions of the model (Sect. 5.3). In addition, we discuss the generalizability and the application of the evaluated solution techniques in real-life markets (Sect. 5.4).

### Main results and insights

Our comparison of different data-driven DP and RL solution techniques showed that the performance and, in turn, the suitability of different methods depend on the available amount of training data as well as requirements regarding computational time. Note, such a comparison was not possible with previous studies, e.g., Zhu et al. (2024) and Groeneveld et al. (2024), considering RL algorithms only.

In our experiments, we observed that already about 10 episodes (200 periods) were enough for DPest to reach at least 76% of the optimal solution. This amount of data was sufficient to approximate the underlying dynamics of the environment and to obtain decent policies via DP without having full knowledge of the market in advance. The example showed that DPest is able to deliver more reliable results in environments with scarce training data than established RL techniques as they require more training data to perform well. With more than 100 episodes of data (i.e., 2000 periods), however, RL algorithms like the on-policy algorithm PPO were able to outperform DPest. With 1000 episodes of data (i.e., 20,000 periods), off-policy RL algorithms like DDPG and TD3 performed best among the examined methods.

Naturally, the observed critical amounts of data that characterize ranges where certain algorithms perform better than other will vary with different model parameters as well as demand definitions, regression features, etc. Nevertheless, for the monopoly experiments and the duopoly experiments, we overall observed similar patterns and relations.

Overall, we find that the most critical aspects that are responsible for the performances of the methods are the following. First, the less components of the environment have to be estimated the better DPest works. If parts of the environment are completely unknown, RL algorithms are likely to perform better than DPest. Second, if single decisions are critical or policies have to be explainable, than RL algorithms should be used with care. Third, the data availability is crucial. It matters whether additional observations can be widely explored or not. Fourth, as long as the problem size is such that DP is still tractable, there are good reasons (cf. transparency, interpretability, trust) to use established approaches like DPest. If not, and if heuristic solutions have to be used anyway, then state-of-the-art RL techniques seem clearly better suited than classical ADP methods. In the considered pricing problems, particularly PPO turned out as a promising alternative, as it provided high performance and fast runtimes.

### Limitations

Naturally, our results of how the different algorithms' performances relate to each other for different amounts of data will not hold in general and depend on specific parameter choices. In this regard, precise practical guidelines for when to use which algorithm cannot be inferred.

Moreover, the scalability of our simulation framework in terms of bigger market sizes and multiple competitors is a limitation. In our current (Python) implementation, the training of all algorithms and their simulation in monopoly markets with larger $|A|$, $|S|$, and $T$ can require runtimes of several hours. The extension of the market size with one or more competitors prolongs these runtimes by the factor of 2 and more. Hence, more realistic scenarios with an oligopolistic character and marketplace sizes of about $T \geq 150, |A| \geq 100$, and $|S| \geq 200$ (the length of the booking period, the price range, and the seating capacity of a regular flight) will require more computational resources than used in our experiments[2] and future work on runtime efficiency. Furthermore, boundaries in terms of state space sizes—in which tabular benchmark methods like QL, DPopt, and DPest are not applicable anymore—were not further investigated within the scope of this work.

Moreover, our basic market model could be further extended. For example, non-myopic customers could be included in the demand model as well as seasonal effects. Additionally, a multi-class system with economy and business class could be introduced.

### Future work directions

In this context, there are several directions for future research, which can further improve the model and address its limitations, cf. Sect. 5.2. First, more complex problem settings, such as, e.g., strategic consumer behavior, network RM, or changing environments (cf. Gatti Pinheiro et al. (2022)), could be investigated. Note, in such problems, DP methods are likely not tractable anymore and alternatives as RL are needed. Second, to address the computational limitations, more efficient implementations, scalable

[2] Experiments A and B—with all repeated training and evaluation runs—were run on a Mac Book Pro with an M1 CPU, 390 MiB RAM were used.

techniques, as well as relaxation or approximation techniques could be used.

Third, the methods compared and the results observed may also be helpful in the selection of an algorithm for other RM applications, such as, e.g., e-commerce or other service industries. Fourth, another promising direction for future research is to study how to (i) best calibrate a digital twin environment, i.e., to estimate consumer behavior and competitors' price reactions from limited observable historic data and to (ii) further examine if high-performing RL methods like DDPG, TD3, and SAC can be suitably pre-trained on digital twins to be used in real-world applications.

## Generalizability and application in practice

As the tested DP and RL algorithms are of a general nature, they can also be applied in further revenue management applications that can be expressed as an MDP. For both types of methods, it is necessary to calibrate the market environment to the specific use case under consideration. In case (components) of the environment are not known, they have to be either estimated from data or approximated by a suitable model. This can often be achieved based on domain knowledge and industry experts.

Alternatively, the market environment has to be suitably defined, e.g., by using historical data. Besides basic market parameters like length of the booking period, price ranges, and (inventory) capacity, the consumers' demand probabilities under competition need to be estimated. Our work showed that this can be appropriately achieved without vast amounts of data, cf. Sect. 4.2, in which we estimated demand. In environments with moderate market sizes, DP methods like DPest could be applied immediately. In more complex scenarios with larger marketplace sizes, RL techniques can be pre-trained on the fitted environment, see, e.g., Groeneveld et al. (2024), without risking real-world revenue losses; see also the recent survey by Rolf et al. (2023).

Which algorithms perform best in other applications can, in general, not be foreseen. However, our methodology to compare different algorithms for basic test problems remains applicable. In this context, we may already infer to skip approaches like ADP or tabular QL. Moreover, our analysis suggests to particularly test DPest for few data cases and look for critical amounts of data, where, e.g., PPO and other RL methods take over.

## Conclusion

In this paper, we examined the performance of various state-of-the-art RL algorithms as well as DP methods with fitted environments in different finite horizon dynamic pricing airline-inspired test markets with limited inventory.

We find that in scenarios with scarce training data, DP methods on fitted environments (DPest) are highly competitive, as on the one hand, few data can be enough to sufficiently estimate the model dynamics, and on the other hand, these data are not enough for RL algorithms to find good policies. In our numerical experiments for monopoly and duopoly examples, we obtain that this critical amount of data—where RL, i.e., in our cases PPO, starts to outperform data-driven DP methods—is between 10 and 100 episodes.

Providing more training data allows off-policy RL algorithms like DDPG and TD3 to perform best among all examined methods, although also other RL algorithms like PPO, A2C, and SAC, as well as the data-driven DP approach DPest obtain similar average rewards.

Comparing the computation times, we obtain that PPO, A2C, and DQN required significantly less training time compared to TD3, DDPG, and SAC, which makes them more suitable for longer training runs.

In summary, for applications with few available data, required transparency of computed solutions, and broad domain knowledge about the model dynamics, established DP methods still seem the best choice. Instead, for applications with more data, less critical single decisions, and fairly unknown environments, RL techniques are a promising alternative to DP-based heuristics as ADP. Notably, among all RL algorithms, PPO appeared as the candidate that best combines fast learning, high rewards, and low computation time.

## Appendix

### Tables of hyperparameters

See Tables 8, 9, 10, 11, 12, 13, and 14.

**Table 8** Hyperparameters for Q-learning (QL)

| Parameter | Value |
|---|---|
| Learning rate | 0.8 |
| Exploration rate | 0.5 |
| Learning rate decay factor | 0.9999 |
| Exploration rate decay factor | 0.999 |
| Learning rate floor | 0.01 |
| Exploration rate floor | 0.1 |

**Table 9** Hyperparameters for deep Q-networks (DQN)

| Parameter | Value |
|---|---|
| Learning rate | $10^{-3}$ |
| Experience buffer size | $10^6$ |
| Steps until learning starts | 100 |
| Minibatch size | 32 |
| Polyak coefficient ($\tau$) | 1.0 |
| Discount factor ($\gamma$) | 0.99 |
| Train frequency | 4 steps |
| Target network update interval | $10^5$ |
| Reduced exploration fraction | 0.1 |
| Initial exploration rate | 1.0 |
| Final exploration rate | 0.05 |
| Maximum gradient clipping | 10 |

**Table 10** Hyperparameters for deep deterministic policy gradient (DDPG)

| Parameter | Value |
|---|---|
| Learning rate | $10^{-3}$ |
| Experience buffer size | $10^6$ |
| Steps until learning starts | 100 |
| Minibatch size | 256 |
| Polyak coefficient ($\tau$) | 0.005 |
| Discount factor ($\gamma$) | 0.99 |

**Table 11** Hyperparameters for twin delayed DDPG (TD3)

| Parameter | Value |
|---|---|
| Learning rate | $10^{-3}$ |
| Experience buffer size | $10^6$ |
| Steps until learning starts | 100 |
| Minibatch size | 256 |
| Polyak coefficient ($\tau$) | 0.005 |
| Discount factor ($\gamma$) | 0.99 |
| Policy delay | 2 |
| Target policy noise | 0.2 |
| Maximum target policy noise clipping | 0.5 |

**Table 12** Hyperparameters for advantage actor critic (A2C)

| Parameter | Value |
|---|---|
| Learning rate | $7 \times 10^{-4}$ |
| Steps per update | 5 |
| Discount factor ($\gamma$) | 0.99 |
| Generalized advantage estimator factor ($\lambda$) | 1.0 |
| Maximum gradient clipping | 0.5 |
| Entropy coefficient | 0.0 |
| Value function coefficient | 0.5 |
| RMSProp optimizer epsilon ($\epsilon$) | $10^{-5}$ |

**Table 13** Hyperparameters for soft actor critic (SAC)

| Parameter | Value |
|---|---|
| Learning rate | $3 \cdot 10^{-4}$ |
| Experience buffer size | $10^6$ |
| Steps until learning starts | 100 |
| Minibatch size | 256 |
| Entropy coefficient ($\alpha$) | Automated |
| Polyak coefficient ($\tau$) | 0.005 |
| Discount factor ($\gamma$) | 0.99 |

**Table 14** Hyperparameters for proximal policy optimization (PPO)

| Parameter | Value |
|---|---|
| Learning rate | $3 \times 10^{-4}$ |
| Steps per update | 2 048 |
| Minibatch size | 64 |
| Epochs per update | 10 |
| *clip_range* ($\varepsilon$) | 0.2 |
| Discount factor ($\gamma$) | 0.99 |
| Generalized advantage estimator factor ($\lambda$) | 0.95 |
| Entropy coefficient | 0 |
| Value function coefficient | 0.5 |

**Funding** Open Access funding enabled and organized by Projekt DEAL.

## References

Bellman, R. 1957. *Dynamic programming*. New York: Dover Publications.

Bertsimas, D., and S. De Boer. 2005. Simulation-based booking limits for airline revenue management. *Operations Research* 53 (1): 90–106.

Bondoux, N., et al. 2020. Reinforcement learning applied to airline revenue management. *Journal of Revenue & Pricing Management* 19: 332–348.

Brockman, G. et al. 2016. Openai gym. *arXiv preprint* arXiv:1606.01540.

Calvano, E., et al. 2020. Artificial intelligence, algorithmic pricing, and collusion. *American Economic Review* 110 (10): 3267–3297.

Chatterjee, K., and T. A. Henzinger 2008. Value iteration. In *25 years of model checking: history, achievements, perspectives*, pp. 107–138. Springer.

Chen, M., and Z.-L. Chen. 2015. Recent developments in dynamic pricing research: Multiple products, competition, and limited demand information. *Production and Operations Management* 24: 704–731.

Collins, A.J., and L.C. Thomas. 2012. Comparing reinforcement learning approaches for solving game theoretic models: A dynamic airline pricing game example. *Journal of the Operational Research Society* 63 (8): 1165–1173.

Collins, A., and L. Thomas. 2013. Learning competitive dynamic airline pricing under different customer models. *Journal of Revenue & Pricing Management* 12: 416–430.

den Boer, A. V., J. M. Meylahn, and M. P. Schinkel. 2022. Artificial collusion: Examining supracompetitive pricing by q-learning algorithms. *Amsterdam Law School Research Paper* (2022-25)

Famil Alamdar, P., and A. Seifi. 2024. A deep Q-learning approach to optimize ordering and dynamic pricing decisions in the presence of strategic customers. *International Journal of Production Economics* 269: 109154.

Fujimoto, S. et al. 2018. Addressing function approximation error in actor-critic methods. *https://arxiv.org/abs/1802.09477*

Gallego, G., and G.J. van Ryzin. 1994. Optimal dynamic pricing of inventories with stochastic demand over finite horizons. *Management Science* 40: 947–1068.

Gatti Pinheiro, G., et al. 2022. Demand change detection in airline revenue management. *Journal of Revenue & Pricing Management* 21 (6): 581–595.

Gerpott, T.J., and J. Berends. 2022. Competitive pricing on online markets: A literature review. *Journal of Revenue & Pricing Management* 21 (6): 596–622.

Gosavi, A., et al. 2002. A reinforcement learning approach to a single leg airline revenue management problem with multiple fare classes and overbooking. *IIE Transactions* 34 (9): 729–742.

Groeneveld, J., et al. 2024. Self-learning agents for recommerce markets. *Business & Information Systems Engineering* 66: 441–463.

Guerrini, A., et al. 2023. Personalization @scale in airlines: Combining the power of rich customer data, experiential learning, and revenue management. *Journal of Revenue & Pricing Management* 22: 171–180.

Haarnoja, T. et al. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML 2018, 10–15, 2018*, volume 80 of *Proceedings of machine learning research*, pp. 1856–1865. PMLR.

Huang, S., A. Kanervisto, A. Raffin, W. Wang, S. Ontañón, and R. F. J. Dossa 2022. A2C is a special case of PPO. *https://arxiv.org/abs/2205.09123*

Isler, K., and H. Imhof. 2008. A game theoretic model for airline revenue management and competitive pricing. *Journal of Revenue & Pricing Management* 7: 384–396.

Kastius, A., and R. Schlosser. 2022. Dynamic pricing under competition using reinforcement learning. *Journal of Revenue & Pricing Management* 21: 50–63.

Klein, R., et al. 2020. A review of revenue management: Recent generalizations and advances in industry applications. *European Journal of Operational Research* 284: 397–412.

Kong, X., D. Kong, J. Yao, L. Bai, and J. Xiao. 2020. Online pricing of demand response based on long short-term memory and reinforcement learning. *Applied Energy* 271: 114945.

Kumari, A., and M. Kumar. 2024. Dynamic pricing: Trends, challenges and new frontiers. In *2024 IEEE international conference on contemporary computing and communications (InC4)*, vol. 1, pp. 1–7.

Lagoudakis, M.G., and R. Parr. 2003. Least-squares policy iteration. *The Journal of Machine Learning Research* 4: 1107–1149.

Lawhead, R.J., and A. Gosavi. 2019. A bounded actor-critic reinforcement learning algorithm applied to airline revenue management. *Engineering Applications of Artificial Intelligence* 82: 252–262.

McGill, J., and G. van Ryzin. 1999. Revenue management: Research overview and prospects. *Transportation Science* 33: 233–256.

Mnih, V., et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518 (7540): 529–533.

Mnih, V., et al. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PMLR

Nataraj, S., et al. 2024. Transfer learning to scale deep Q networks in the context of airline pricing. *Journal of Revenue & Pricing Management, to appear,*. https://doi.org/10.1057/s41272-024-00493-7.

Powell, W.B. 2007. *Approximate dynamic programming: Solving the curses of dimensionality,* vol. 703. New York: Wiley.

Raffin, A., et al. 2021. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research* 22 (268): 1–8.

Rolf, B., et al. 2023. A review on reinforcement learning algorithms and applications in supply chain management. *International Journal of Production Research* 61 (20): 7151–7179.

Schlosser, R., and M. Boissier. 2018. Dealing with the dimensionality curse in dynamic pricing competition: Using frequent repricing to compensate imperfect market anticipations. *Computers & Operations Research* 100: 26–42.

Schlosser, R., and K. Richly. 2019. Dynamic pricing competition with unobservable inventory levels: A hidden Markov model approach. In *Operations Research and Enterprise Systems, CCIS 966*, ed. G.H. Parlier, F. Liberatore, and M. Demange, 15–36. Springer.

Schulman, J., et al. 2017. Proximal policy optimization algorithms. arXiv:1707.06347

Selcuk, A.M., and Z.M. Avsar. 2019. Dynamic pricing in airline revenue management. *Journal of Mathematical Analysis & Applications* 478 (2): 1191–1217.

Shihab, S.A.M., and P. Wei. 2022. A deep reinforcement learning approach to seat inventory control for airline revenue management. *Journal of Revenue & Pricing Management* 21: 183–199.

Si, J., A.G. Barto, W.B. Powell, and D. Wunsch. 2004. *Handbook of learning and approximate dynamic programming*, vol. 2. New York: Wiley.

Silver, D., et al. 2014. Deterministic policy gradient algorithms. In: *ICML'14*, vol. I, pp. 387–395.

Smith, B.C., et al. 1992. Yield management at American airlines. *Interfaces* 22 (1): 8–31.

Sniedovich, M., and A. Lew 2006, 01. Dynamic programming: An overview. *Control and Cybernetics 35*.

Sutton, R. S., and A. G. Barto. 2018. *Reinforcement learning—An introduction*, 2nd ed. Adaptive computation and machine learning. New York: MIT Press

Talluri, K., and G. van Ryzin. 2004. Revenue management under a general discrete choice model of consumer behavior. *Management Science* 50: 15–33.

Watkins, C.J., and P. Dayan. 1992. Q-learning. *Machine Learning* 8: 279–292.

Yavuz, T., and O. Kaya. 2024. Deep reinforcement learning algorithms for dynamic pricing and inventory management of perishable products. *Applied Soft Computing* 163: 111864.

Zhou, Q., et al. 2023. Joint pricing and inventory control with reference price effects and price thresholds: A deep reinforcement learning approach. *Expert Systems with Applications* 233: 120993.

Zhou, Q., Y. Yang, and S. Fu. 2022. Deep reinforcement learning approach for solving joint pricing and inventory problem with reference price effects. *Expert Systems with Applications* 195: 116564.

Zhu, X., L. Jian, X. Chen, and Q. Zhao. 2024. Reinforcement learning for multi-flight dynamic pricing. *Computers & Industrial Engineering* 193: 110302.