Dynamic Pricing using Reinforcement Learning in Hospitality Industry

Inderpreet Singh
Thoughtworks
Inder.sk16@gmail.com

***Abstract*— Hotel room pricing is a very common use case in the hospitality industry. Such use cases take dynamic pricing strategies for setting optimum prices wherein prices are dynamically adjusted based on user engagement. However, it is challenging to design an approach that makes pricing dynamic with respect to complex market change. In this paper, we suggest a reinforcement learning based solution for this problem. The approach employs a Deep Q-Network (DQN) agent trained to recommend/suggest optimum pricing strategies which maximizes the total profits for a day. In addition, the pricing strategy is optimized in such a way that empty rooms remain minimal. A real-life hotel-bookings data set is being used for testing this approach. The data is aggregated and preprocessed before being used for the task. The pricing strategy is influenced by the hotel-demand, type of rooms, number of nights and other variables. The hotel-demand is derived from a Random-forest model trained on the processed data to simulate original demand distribution of processed data. Using the DQN based dynamic pricing strategy, a potential 15-20 percentage higher reward(profits) were obtained compared to fixed pricing, and rule-based pricing strategy. At the same time the empty rooms left were significantly lower for the DQN based approach.**

Keywords*— *Dynamic Pricing, Deep Reinforcement Learning, Hospitality Industry, Deep Q-network, Revenue management

I. Introduction

Dynamic pricing is a popular pricing strategy used widely in retail use-cases in stores. Such use cases have been researched upon widely [1][2]. Moreover, Travel industry also has use cases involving dynamic pricing upon which some research has been done [3].

Such strategies are employed in the Hospitality Industry too, in use-cases such as hotel pricing, overbooking control, etc. However, the research on dynamic pricing strategies/models for the hotel/hospitality industry has been very limited. Historically hotel revenue management systems have been used for dynamic pricing strategies [4][5]. These systems are quite primitive given they are mostly rules based and do not involve any machine learning.

In recent times, research has been done for dynamic pricing use cases in hotel revenue management which involves using linear and nonlinear stochastic models involving Markov Chain Monte Carlo (MCMC) models [6][7]. Such strategies have shown good results. However, these strategies also have certain demerits. Such strategies cannot be scaled if additional factors need to be accounted for, these strategies are quite generalized and lack the scope of being personalized and do not give any insights on user behaviour.

In our paper, the solution involves using a reinforcement learning approach where we train a deep Q-network agent to learn the pricing strategy so as to maximize the profits while minimizing the empty rooms left at the same time. The motivation for such an approach comes from its proven effectiveness in dynamic pricing use cases in retail domain [8]. Moreover, this approach solves the problems with previous works like scalability, models can be trained for personalized use cases. In addition to that, reinforcement learning based models can be trained in online fashion without even the need for historical data.

In the case of hotels, pricing strategies are governed mainly by the demand and choices the customer makes while selecting the rooms. The general behaviour is as follows:

- Hotels increase their pricing if/when they observe high demand
- Having additional services along with the room also leads to higher prices
- Time of booking also impacts pricing, bookings made during special days (christmas, new year) leads to higher price
- Bookings made in advance are generally economical
- Competition pricing also impacts the pricing for a hotel
- Other features such as type of rooms, size of rooms, stay nights, etc affect the pricing decisions too

In this paper, the reinforcement learning approach learns pricing strategies using a Deep Q-Network(DQN) agent considering the impact of all the mentioned variables except the competition pricing. Competition pricing can lead to complex effects. Such behaviours are complex to handle and are out of scope for this paper. Unlike the historical dynamic pricing approaches which use a demand simulation function[9] we have used a real life hotel-bookings data set. We have trained a machine learning model on the processed hotel-bookings data. This machine learning based demand calculation approach is quite capable given it can be scaled to use new variables that impacts the demand. It also has the advantage of transferability, such architectures without much updating can be used for approximating demand for a different geography, context.

The remaining paper is organized as follows: Section 2 lists many previous work related to dynamic pricing strategies across multiple industries. Section 3 introduces the methodology created for hotel pricing scenario, where a life cycling pipeline was build from user demand forecasting, action and environment define to agent deep Q-network implementation. Discrete pricing model and offline model training will be discussed as well. Results regarding our experiments on hotel use case are in section 4, which validate the effectiveness compared with the traditional pricing approaches.

978-1-6654-9291-1/22/$31.00 ©2022 IEEE

II. Previous Work

Dynamic Pricing(DP) has been used and researched widely across different domains [10][11]. Retail industry was the first to benefit from this, historically researchers have suggested different approaches for DP in retail use cases. Most of these assume demand to follow a distribution (poisson, constant Rate distribution)[12], other systems regard demand as a polynomial function dependent on price and product expiry [13][14].

Another major industry where DP powers the revenue management systems is the airline and travel industry. Reference [3] suggests multiple approaches to DP in the airline industry, each being an ensemble of a demand forecasting system (linear model or deep neural networks) followed by a logistic price optimization function.

We have used a reinforcement learning based deep Q-network approach for DP problems for our hotel room pricing use case. Reinforcement learning based Q-learning systems have shown to be great at price optimizations [15]. However, Reinforcement learning based Q-learning optimization systems do scale very well for large action space [16]. Using deep Q-network based systems removes the need for maintaining a Q-learning table which can cause memory issues. Moreover, Deep Q-networks are quite customize and scalable as well

In the past mostly the Reinforcement learning based DP systems have centred around single agent setup. However, there has been some research done for multi agent price optimization problems [16][17][18]. Our deep Q-network solution trains a single agent to learn price variations, also it maximizes the profits across all customers for a fixed time interval of 1 month. Researchers have also worked on personalized DP problems where profit maximization happened at each user/customer level [19][20]

Our solution solves DP for hotel room pricing. In today's time, DP is a very popular technique used in the hotel and hospitality industry. Moreover, a sizable research has happened for DP in hotel pricing related use cases [6][21]. Ref.[22] have used nonlinear programming for training the agent on historical demand data for price optimization. Ref.[23] compared the pricing strategy across 1000 hotels in Europe. Factors such as additional amenities (parking, etc), booking agent, number of days booking was made before arrival were found to be impacting price along with demand elasticity.

Discrete price points have been used for DP optimization problems since using a continuous set of prices increases the action space exponentially[24]. Most of the previous work on DP for hotel industry used predefined probability distributions to represent bookings [25]

III. Methodology

In our study, the life-cycling process has been divided into four major steps, each step relies on the feedback from the previous step and vice versa.

Firstly, it is necessary to dive deep into the real market scenario to identify certain product demands from customers in the market. The feedback from the market plays an important role in setting up a comprehensive environment where Reinforcement Learning(RL) is able to take effect. Secondly, we defined and customized the basic features of RL, including environment, state, reward and action. Once market demands understanding and environment setup is ready, then it comes to the agent (deep learning model) configuration and exploration/exploitation resolve, which are core components of Reinforcement Learning.

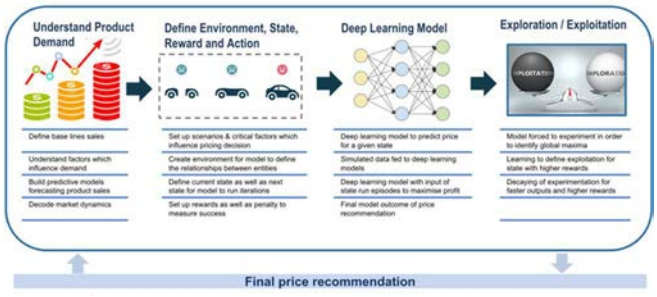


Fig. 1. Major Process of Dynamic Pricing using RL

A. Product Demand Forecasting

Understanding the product (hotel rooms demand) is the first building block for the RL. Theoretically, a demand simulation function is required to simulate demand which is dependent on your environment state and action space.

Pricing optimization problems such as dynamic pricing tries to learn price as a function of price dependent variables. Demand variation is a key parameter which impacts the price. Generally demand is estimated / simulated as a distribution governed by some polynomial or exponential function. However, since our solution trains in an offline manner therefore we have used a real dataset for hotel bookings. Instead of simulating the demand function, we have trained a machine learning model for predicting demand/bookings for a combination of price, date, etc. using the hotel data.

Using the ML model helped us capture complex multivariate dependencies of demand, also it allowed us to model each variable as a non-parametric function with non-linear dependency on demand

As a part of our experimentation, we tried modeling demand using multiple combinations of variables at different levels of aggregations. key variables used for demand modeling are:

Lead-time: Number of days that elapsed between booking date and arrival date

Hotel Agent: ID of the travel agency that made the booking

Stay nights: number of nights spent in hotel

Stay Date: Date of stay

Hotel type: Type of hotel for which booking is made

Room type: Code for the type of room assigned to the booking

Also, we experimented with Linear regression derivatives and tree based non-parametric ensemble regressors (Catboost, etc.). Based on the performance indicators (Loss metrics) and model robustness (out of context / time validation) scores, Random forest regressor was used for demand prediction using different levels of aggregations for the above-mentioned variables.

TABLE I. Demanding Model Comparisons Using Different Architecture

Architecture	Mean Absolute Error	MAPE(ratio)
Random Forest	0.5	0.18
Linear Regression	1.18	0.34
Catboost Decision	0.62	0.2
Tree Regressor	0.68	0.21

B. Define Environment, State, Reward and Action

The idea of Reinforcement Learning is the interaction between agent (RL model) and environment, agent keep updating via taking rewards generated from environment. In our use case, reward refers to the total profit of the hotel.

Basically, there are five features which consist of the Reinforcement Learning life-cycle. We will discuss environment, state, reward and action sequentially except the agent, which will be illustrated in the next part independently.

Environment

In our user case, the environment could be considered as a real hotel room category. Typically the hotel has a fixed initial number of rooms at the beginning and no additional rooms can be allocated during the selling period. We define one selling period (or one episode) as the time when the room is available for booking (typically one year ahead of stay date), to the time of stay date post which rooms are vacant and leading to a loss for the hotel. Hotel booking comes with additional add-ons like meal plan, room nights, agent platform, all those add-ons together jointly determine the pricing strategy in our user case. Hotel demands from customers were represented by the number of bookings at a particular stay date given a particular combination of add-ons, while booking numbers were simulated based on the demand model we discussed in the last section.

Fig. 2. Reinforcement Learning Mechanism

State

The states are descriptions of environment feedback, it is formulated as an eighteen element vector St = [LT, inventory,combo1 bookings, combo2 bookings, ..., combo16 bookings]. Lead Time (LT) is a terminology that illustrates the gap between booking date and exact arrival date or stay date. Usually it ranges from a year before arrival date to a day before, cut off ranges are widely used to determine the level of lead time such as how close the booking date is from arrival date since the earlier booking, the lower price based on common sense. In our scenario, we configure the cut off criteria of lead time based on Table III. The price action is selected based on the current state. When a price action is applied, the state is then transited to the next state St +1= [LT-1, new inventory, . . .]

Ex.

combo1: 4 room nights + Agent Expedia + Lead Time 1

combo2: 7 room nights + Agent Booking.com + Lead Time 2

Each combo represents a particular set of selected room nights, agent and lead time. 16 combos in total. To replicate real scenarios, not all combos will always have bookings(random sampling).

TABLE II. STATES

Current State	Remaining Occupancies	Lead Time	Combo1 booking as num	Combo2 booking as num	...
S(t)	100	10	2	1	...
S(t+1)	98	9	2	5	...

TABLE III. LEAD TIME CUT OFF TABLE.(SUPPOSE A GUEST INTENDS TO ARRIVE ON DEC 25TH, 2016)

Start Date	End Date	Duration(Days)	Lead Time Level
2016-12-21	2016-12-25	4	1
2016-12-12	2016-12-21	13	2
2016-11-26	2016-12-12	29	3
2016-11-08	2016-11-26	47	4
2016-10-14	2016-11-08	72	5
2016-09-14	2016-10-14	102	6
2016-08-06	2016-09-14	141	7
2016-06-22	2016-08-06	186	8
2016-04-03	2016-06-22	266	9
before 2015-12-25	2016-04-03	>=366	10

Reward

The reward is total profit with waste and discount being taken into account, a cost will be added if remaining occupancies are larger than zero by the end of the lead time period.

$$r_{t+1} = price * (1 - RoomNightDiscount) * (1 - AgentDiscount) * sale_t - cost * waste \quad (1)$$

Action

In RL, by performing actions, the agent transits between different states. Ideally, a good RL model should choose suitable actions from the actions space which can maximize the reward.

In our case, an action space was created as a look-up table consisting 1028 combinations of price(max) and discounts.

Each action in action space(1028 size) is an array

$$[D_{rn1}, D_{rn2}, D_{rn3}, D_{rn4}, D_{a1}, D_{a2}, D_{a3}, D_{a4}, P_{max}] \quad (2)$$

Where

$D_{rn1}, D_{rn2}, D_{rn3}, D_{rn4}$ *- discounts associated with each of 4 values of room-night*

$D_{a1}, D_{a2}, D_{a3}, D_{a4}$ *- discounts associated with each of 4 values of agent*

P_{max} *- max price (on which discounts are to be applied)*

$P_{max} \subset p_1, p_2, p_3, p_4, p_5$ *: 5* P_{max} *values in pricing space*

$D_{rn(i)} \subset d(i)$, *0 : either 0 discount or d(i) discount - different for each room-night*

$D_{a(i)} \subset d(i)$, *0 : either 0 discount or d(i) discount - different for each agent*

C. Agent

Agent is the core component of Reinforcement Learning; it observes the states and rewards from the environment and then acts to optimize accordingly. In our study, an agent is a deep learning model that uses deep neural networks as architecture to play with.

Discrete Pricing Action Models

Discrete reinforcement learning model means the action space is discrete, in our case it means the pricing strategy is a discrete list [$price_1$, $price_2$, . . .], the agent picks one of the best from this list. We used Deep Q-Learning Network (DQN) as the architecture. DQN accepts states as input and returns Q values via neural network flow, Q values will be used to look up action needs to be taken from action space.

In Q-learning, the Q values are updated following the Bellman optimality equation:

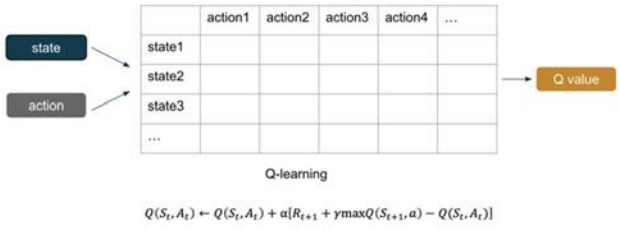
$$Q(S_t, a_t) \leftarrow Q(S_t, a_t) + \alpha * (r_{t+1} + \gamma * maxQ(S_{t+1}, a) - Q(S_t, a_t)) \quad (3)$$

In the case of neural network, the model is trying to minimize the difference between these two values,

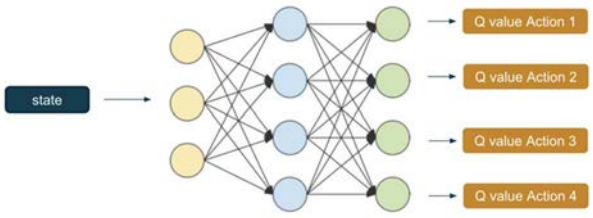
$$Loss: [Q(S_t, a_t), Q(S_t, a_t) + \alpha * (r_{t+1} + \gamma * maxQ(S_{t+1}, a) - Q(S_t, a_t))] \quad (4)$$

It is common to choose the MSE loss for simplicity and computation efficiency, in some cases, smooth-l1 loss can also be used as it's less sensitive to outliers and it also prevents exploding gradients to some extent. The is the learning rate for Q and is set to be 1. Note it is different from the learning rate of the network optimizer which is usually set at a small number between 1e-6 to 1e-2. The discount factor is a number between 0 and 1. It essentially determines how much the agent cares about the immediate reward versus the rewards in the distant future. In the case of $\gamma = 0$, the agent is completely shortsighted and cares only about the immediate reward. If $\gamma = 1$, the agent is foresighted and evaluates all future rewards for each action. In practice, we set between 0.9 and 0.99.

DQN doesn't simply replace the state-action value table with a neural network, to ensure training stability in deep neural networks, a replay memory buffer is introduced in the DQN framework. The replay memory buffer stores the sequential transitions (S_t, a_t, r, S_{t+1}) and during training, mini-batch samples are randomly chosen from memory. The reason is that the sequential transitions are usually correlated, by storing large enough transitions into memory and then resampling can help get independently distributed samples for the neural network.



(a) Q-learning



(b) Deep Neural Network Architecture

Fig. 3. (a) Q-Learning shows that Deep Q-Learning Network accepts states from environment as input and output the Q-Values. sub-figure(b) illustrates the specific architecture of deep neural network, we used a simple Feedforward Neural Network with 3 hidden layers, the number of neurons are 200, 100, 50 accordingly. Input layers are the states while output layers are action spaces with size 1024

The other main innovation in DQN is the involvement of the target network. At the early age of DQN, there was only one network which was used for both predicting and targeting. That means the learning target moves simultaneously with the parameters that are trying to learn. Though a single policy network could work in some cases, it is like tracing a moving target and sometimes causes the failure of RL. The target network is introduced to maintain the stability of the learning process. It is a lagged copy of the policy network and used specifically to calculate Q-values for the loss function (or target). During the training, the policy network is updated each step, while the target network is updated slowly (soft update) or copies policy network every r step (hard update).

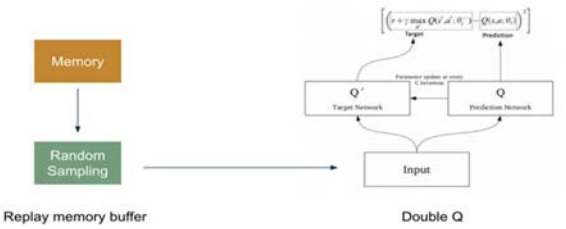


Fig. 4. Replay Memory Buffer

The details of discrete pricing algorithm is sketched in Algorithm 1.The hyper-parameters used in Algorithms are described in Table IV.

Algorithm 1 DQN-based pricing algorithm

```
Initialize storage memory to size N
Initialize action-value function Q (policy network) with random
weights w
Initialize target action-value function Q (target network) with same
weights w
for i = 1 to M do
   Reset state to initial state values (CASE: [LT, remaininginventory, 0, 0, ..., 0])
   for t = 0 to T do
      execute action using ε − greedy;
      calculate demand based on price, time, and discount;
      calculate sales based on demand and inventory;
      calculate reward r_{t+1};
      calculate remaining inventory, i.e., inventory - sale;
      calculate next state S_{t+1};
      store transitions (S_t, a_t, r_{t+1}, S_{t+1}) in memory;
      randomly sample mini-batch of transitions (S_j, a_j, r_{j+1}, S_{j+1}) from
memory;
      calculate target Q: if end of episode, then r_{j+1}, otherwise,
follows Bellman optimality equation using target network (w);
      perform a gradient descent step on loss (Q(s_j, a_j; w), Q) with
respect to the policy network (w);
      update target network w = w every r steps (hard update) or update
target network w = τ * w + (1 − τ) * w (soft update);
   end for
end for
```

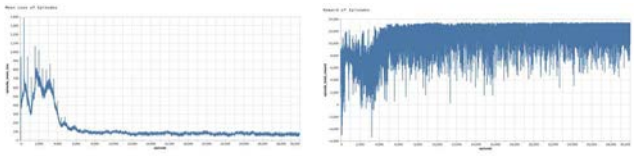
TABLE IV. HYPERPARAMETERS

Symbol	Parameter Name	Values
M	Number of episodes	30k
α	Optimizer learning rate	1e-6 - 1e-2
T	Training step in one episode	10
N	Memory size	5000
γ	Discount factor	0.9
τ	Soft update rate for target network	0.01
r	Hard copy frequency for target network	400
ϵ	Probability of a random action in -greedy method	0.005-0.5
n	Mini-batch size	512

Offline Training

Offline training is the way we used to train a RL model for experiments. Offline means the model will keep being trained and optimized until being deployed to the customer environment, no in-time market feedback is able to impact and update model performance. We trained RL models by tuning the hyper-parameters in Table IV. We set number of episodes equal to 30k and memory buffer size to 5000, min-batch size is 512, training step in each episode is equal to 10. It took around three to four hours on CPU (Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz).

We generated a loss curve and reward curve to check DQN model convergence status, as you can see in Fig. 5, both loss and reward will converge after a certain number of episodes. The training episode stops when time reaches Lead Time equal to one (10 Lead Time) or the inventory becomes 0.



(a) Loss Curve (b) Reward Curve

Fig. 5. (a)Loss Curve shows that the MSE loss converged after a certain number of episodes rand (b)Reward Curve shows that the reward reached maximum after a certain number of episodes as well, both curve verify the convergence of Reinforcement Learning Model

D. Exploration-exploitation

Exploration-exploitation dilemma is a common problem that one has to face in RL. On one hand, we want the agency to execute the best action based on the current knowledge. On the other hand, we want the agency to sufficiently explore the environment to find out actions that may return higher rewards. In this project, we used -greedy method to ensure the exploration: the probability to take a random action (explore) and probability $1 - \varepsilon$ to select an action based on learned Q values (exploitation). Here ε is set at a relatively high value in the beginning of the training process, and then decays exponentially to fine tune the policy (so called annealed -greedy).

Ideally, We expect more exploration at the beginning and more exploitation of the model as training goes on. Fig. 6 is the graph that show how the number of actions changed between exploration and exploitation as the episodes are going forward. As you can see, the model is taking action randomly when it comes to the exploration stage (the prices are evenly distributed), while the model is heading towards the price that can maximize the profit once the phase has moved to exploitation.

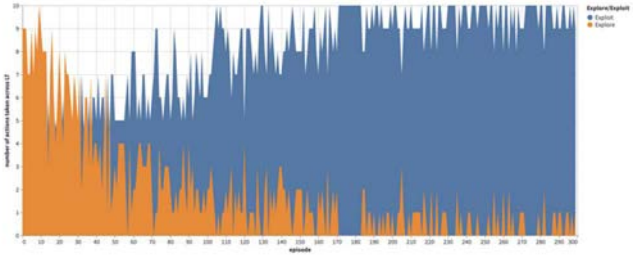


Fig. 6. Explore/Eploit taken actions in training

E. Overall View of Life-Cycling RL Training

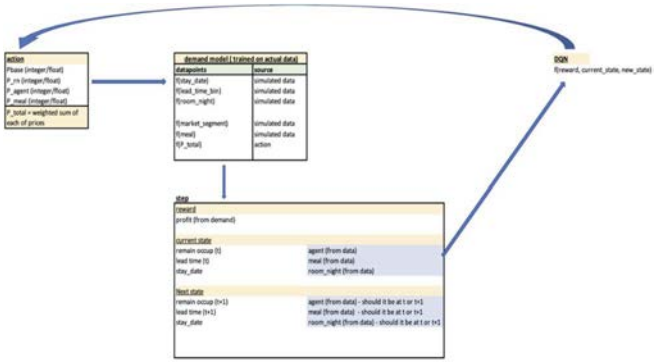


Fig. 7. Life-Cycling DQN Training

Fig. 7 is the diagram illustrating the RL environment. Demand model and DQN represents environment and agent, DQN acts under demand model circumstances, demand model returns new state and update reward to help DQN adjust picking pricing strategy. This dynamic circle will keep moving until the number of episodes requirements has reached.

IV. RESULTS

We used real hotel market data to build a demand model that owns the capability to simulate the demands of customers in the real market. Then we introduced a data simulation approach that would enumerate the hotel bookings combination possibilities at a lead time basis, that is able to help replicate environment and update states.

Pre-trained DQN model was created during the training phase with 30K episodes going through. Then we used this pre-trained DQN model for evaluation with a valid Lead Time (from 10 to 1). Ideally, base price should increase as the arrival date is getting closer due to the higher demand from market, the Fig. 8 depicts the trends of all related metrics including base price, discounts, number of bookings across different lead times.

As you can see there is a slight increase until the final time regarding the base price and number of bookings should have a rough inverse relationship with base price. Intuitively speaking, the inverse relationship between number of bookings and base prices existed in most scenarios, that is aligned with the real market behaviors. As for the huge jump for last step lead time (from lead time 2 to 1), it can be illustrated with the fact that hotel is trying to empty its remaining occupancies via a relatively low pierce as soon as it can before the booking date is close, in that case no wastage

will be resulted. Apart from base price and number of rooms, discounts is zero when lead time level is somewhere middle or demands are at peak, but for the beginning and end of lead time a stimulus bonus are added to drive market demands.

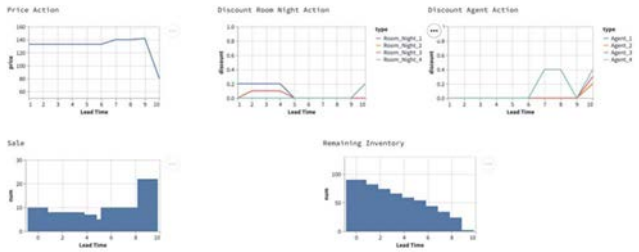


Fig. 8. Summary of Dynamic Pricing(Please note this, the X-axis Lead Time should be viewed reversely, for instance value 0 means Lead time level 10, value 1 means Lead time level 9 so on and so forth)

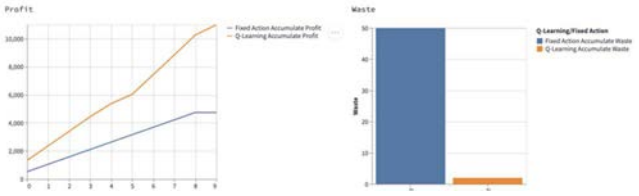


Fig. 9. Fixed Pricing vs Dynamic Pricing

V. Conclusions

We implemented a deep reinforcement learning framework for dynamic pricing on hospitality industry. We first build a machine learning based model for simulating and predicting the market demands using real hospitality dataset.

The result section shows that a significant profit increase will be achieved and less empty rooms remaining if reinforcement learning being applied in dynamic pricing compared to a fixed pricing strategy. Hotel price goes up as the leading time is close to customer expected stay date, which is consistent with common market behavior. For improvements, we proposed three major advancements to make our work more efficient.

- Using twin-DDPG approach to make our action space (pricing strategy) a continuous space
- Exploring the possibility of online training that can let RL models interact with the real time markets demands feedback

References

[1] Ningyuan Chen, G. G, "A primal-dual learning algorithm for personalized dynamic pricing with an inventory constraint", Special Issue:Learning Approaches for Negotiation Agents and Automated Negotiation, 2021.

[2] Ravi Ganti, Matyas Sustik, Q. T. B. S., "Thompson sampling for dynamic pricing", 2018.

[3] Naman Shukla, Arinbjörn Kolbeinsson, K. O. L. M. K. Y, "Dynamic pricing for airline ancillaries with customer context", 2019.

[4] Jiawei Wen, Puya Vahabi, M. G., "Revenue maximization of airbnb marketplace using search results", 2019.

[5] Christine Currie, R C H Cheng, H. K. S., "Dynamic pricing of airline tickets with competition. Journal of the Operational Research Society, 2008.

[6] Aldric Vives, M. J., "Dynamic pricing for online hotel demand: The case of resort hotels in majorca. Vacation Marketing", pp.161–168, 2019.

[7] Abd El-Moniem Bayoumi, Mohamed Saleh, A. A. H. A. A. A. A., "Dynamic pricing for hotel revenue management using price multipliers. Revenue and Pricing Management", 2012.

[8] Roberto Maestre, Juan Duque, A. R. J. A., "Reinforcement learning for fair dynamic pricing", 2018.

[9] Pelin Pekgün, Ronald P. Menich, S. A. P. G. F. F. D. K. M. J. V. S. K. C. J. F., "Carlson rezidor hotel group maximizes revenue through improved demand management and price optimization. INFORMS Journal on Applied Analytics", 2013.

[10] Boer, A. V., "Dynamic pricing and learning: Historical origins, current research, and new directions. Surveys in Operations Research and Management Science", pp.1–18, 2015.

[11] Indre Deksnyte, Z. L., "Dynamic pricing and its forming factors. International Journal of Business and Social Science", pp.1–18, 2012.

[12] Gabriel R. Bitran, S. V. M., "Periodic pricing of seasonal products in retailing. Management Science", pp.43:64–79, 1997.

[13] Youyi Feng, B. X., "A continuous-time yield management model with multiple prices and reversible price changes. Management Science", pp.46:644–657, 2000.

[14] Dina Elreedy, Amir F. Atiya, S. I. S., Novel pricing strategies for revenue maxi- mization and demand learning using an exploration–exploitation framework. Management Science, pp. 25:11711–11733, 2021.

[15] Rupal Rana, F. S. O. "Dynamic pricing policies for interdependent perishable products or services using reinforcement learning". Expert Systems with Applications: An International Journal, pp.42:426–436, 2015.

[16] Erich Kutschinski, Thomas Uthmann, D. P. "Dynamic pricing policies for inter- dependent perishable products or services using reinforcement learning". Journal of Economic Dynamics and Control, pp.27:2207–2218, 2003.

[17] Könönen, V., "Dynamic pricing based on asymmetric multiagent reinforcement learning". Special Issue:Learning Approaches for Negotiation Agents and Automated Negotiation, pp.21:73–98, 2006.

[18] Eduardo R. Gomes, R. K., "The Dynamics of Multiagent Q-Learning in Commodity Market Resource Allocation", volume 263, Springer, 2006.

[19] Yiyun Luo, Will Wei Sun, Y. L., "Distribution-free contextual dynamic pricing", 2021.

[20] Ningyuan Chen, G. G., "A primal-dual learning algorithm for personalized dynamic pricing with an inventory constraint." Special Issue:Learning Approaches for Negotiation Agents and Automated Negotiation, 2021.

[21] Aldric Vives, Marta Jacob, M. P., Revenue management and price optimization techniques in the hotel sector: A critical literature review. Tourism Economics, pp.24:720– 752, 2018.

[22] Muhammad Fadly, Ari Yanuar Ridwan, M. D. A., "Hotel room price determination based on dynamic pricing model using nonlinear programming method to maximize revenue". IEEE, 2019.

[23] Thrane, C., Examining the determinants of room rates for hotels in capital cities: The oslo experience. Journal of Revenue and Pricing Management, pp.5:1287–1309, 2007.

[24] Gadi Fibich, Arieh Gavious, O. L., "The dynamics of price elasticity of demand in the presence of reference price effects", Journal of the Academy of Marketing Science, 2005.

[25] Qing Ding, Panos Kouvelis, J. M. M., "Dynamic pricing through discounts for optimizing multiple-class demand fulfillment", Operations Research, 54:2–200, 2006.