Offline Reinforcement Learning for Price-Based Demand Response Program Design

Ce Xu*, Bo Liu†, and Yue Zhao*
*Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY 11794, USA
†Amazon, Seattle, WA 98109, USA
Emails: {ce.xu, yue.zhao.2}@stonybrook.edu, liubo19831214@gmail.com

2023 57th Annual Conference on Information Sciences and Systems (CISS) | 978-1-6654-5181-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/CISS56502.2023.10089681

***Abstract*—In this paper, price-based demand response (DR) program design by offline Reinforcement Learning (RL) with data collected from smart meters is studied. Unlike online RL approaches, offline RL does not need to interact with consumers in the real world and thus has great cost-effectiveness and safety advantages. A sequential decision-making process with a Markov Decision Process (MDP) framework is formulated. A novel data augmentation method based on bootstrapping is developed. Deep Q-network (DQN)-based offline RL and policy evaluation algorithms are developed to design high-performance DR pricing policies. The developed offline learning methods are evaluated on both a real-world data set and simulation environments. It is demonstrated that the performance of the developed *offline* RL methods achieve excellent performance that is very close to the ideal performance bound provided by the state-of-the-art *online* RL algorithms.**

I. Introduction

The growth of smart buildings, smart homes, and electric vehicles (EVs) has led to great promises in demand response (DR) for achieving a more efficient and sustainable energy system. However, major challenges arise in efficiently extracting DR from a massive number of heterogeneous and often small-scale distributed energy resources (DERs). In particular, there is often a lack of accurate models of such DR resources' dynamic capacity, which often depend on highly variable factors such as human behaviors.

As such, consumer behavior learning and making operation (e.g., control or pricing) decisions accordingly has been an important topic for DR program designers [1]. Naturally, utilities frequently interact with consumers in daily operations by which better understanding of consumers' DR responses may be obtained. Indeed, many DR control policy/program design problems can be modeled as sequential decision-making problems, and Reinforcement Learning (RL) has been studied for solving such problems. Examples include using RL to learn the optimal control of household appliances [2], selecting optimal subset of consumers to call for DR events [3], setting incentive payments to encourage demand reduction [4], and learning the best bidding strategies for DR market participation [5]. In this paper, we focus on the design of price-based DR programs (PBDRP) such as real-time pricing (RTP) [6]. Specifically, we are interested in using dynamic prices as indirect control signals to elicit desired demand responses from consumers.

This work is supported by the National Science Foundation under Grant ECCS-2025152.

Existing works that use RL to design dynamic pricing DR programs have primarily employed *online* approaches. [7] uses online multi-agent RL to learn optimal dynamic pricing strategy and energy consumption schedule that minimize the total cost of service provider and disutility of energy consumers. [8] designs a daily iterative dynamic pricing DR algorithm based on online RL, aiming at cost minimization for both service providers and consumers. In [9], a utility uses online RL to learn consumers' response functions and adjusts its pricing strategies accordingly to achieve certain load reduction target.

One fundamental limitation of such online RL approaches is that the learning process relies on extensive interactions with the consumers. Given that RL is notorious for its slow convergence, it can potentially take months if not years for such online RL approaches to learn a high-performance policy. Moreover, the exploration component of such RL approaches implies that the decision-maker would often need to venture into unseen scenarios which could lead to safety and system reliability risks. In sum, online RL approaches run the risk of being time-consuming, costly and unsafe when applied to learning effective DR control policies/programs with real-world consumers.

To overcome these obstacles, in this paper, we investigate completely *offline* RL approaches that exploit *existing* data sets to discover cost-effective and safe DR policies. Notably, many utilities already have RTP programs for both commercial and residential consumers [10]. As a result, large amounts of consumer load data under RTP programs are already available for many utilities to exploit. Recent studies have investigated the possibility to use offline RL on its own or combined with online learning for demand bidding [11] and device/building control [12]. However, very few works exist that use offline RL for dynamic pricing DR program design. To the best of our knowledge, the only such related work is [13] which aims to reduce energy consumption in office buildings using dynamic pricing. There, online RL is combined with offline pre-training to accelerate convergence. We however note that a) the problem setting therein is very different from ours, and b) as opposed to still partly relying on online RL, we aim to not rely on online RL at all but fully perform offline RL.

In particular, we propose a fully offline RL framework for price-based DR program design. To the best of our knowledge, this is the first work that uses fully offline RL to design dynamic pricing policies for demand response. We first formulate the dynamic pricing design problem as a sequential decision-making problem. Exploiting the availability of smart meter data of consumers, a novel data augmentation method based on bootstrapping is developed to greatly increase the size of the data set for offline RL. We then develop Deep Q-network (DQN)-based offline RL and policy evaluation methods for designing and evaluating DR pricing policies completely based on existing data sets. We conduct extensive numerical experiments to evaluate our proposed offline RL methods. These include evaluations with both a real-world data set and simulation environments. The evaluation demonstrated high performance of the developed methods. Notably, as verified in the simulation environments, the offline learned policies perform very closely to the ideal performance bound achieved by the state-of-the-art online RL algorithms.

II. Problem Formulation and System model

Consider a utility company serving a group of N electricity consumers via PBDRP. Given reliability and/or economic concerns, the utility has a sequence of DR targets over time that it aims to achieve. For this, an effective dynamic pricing policy is desired to encourage the consumers to follow the DR targets as closely as possible. In real-world RTP programs, the electricity prices are typically announced with an advanced notice [10]. In this work, we model such advanced notice as general m-hour ahead notices where m is a program parameter. As such, our problem is to design an *m-hour ahead pricing policy* that can effectively elicit consumer DR responses to follow the desired DR targets.

We model this problem as a sequential decision-making process: a) the state variables are consumer states and certain environmental variables such as weather and time, b) due to the m-hour ahead notice, the control variable/action at time t is the price signal at time $t+m$, and c) the reward at time t is determined by how close the consumers' DR is to the desired DR target at t. Our design objective is a pricing policy that maps states to actions. More details follow.

A. Sequential Decision-Making, MDP, and Delayed Rewards

Given a pricing policy π and some initial state s_0, for each hour i within a day,

- At the beginning of hour i, the utility observes current state s_i, and announce a price signal $a_i \sim \pi$ for hour $i+m$.
- Upon observing the current price a_{i-m} and the announced future price signals $a_{i-m+1}, \dots a_i$, the consumers respond to the price by either decreasing or increasing their loads according to their preferences.
- At the end of hour i, the utility company observes the next state s_{i+1} and observes the reward r_i depending on the customers' responses affected by the price signals.
- Iterate the above process until the end of the day.

We model this decision-making process as a Markov decision process (MDP) represented by a tuple (S, A, r, γ), where:

S is the state space where the state at each hour t is $s(t) = \{t, T(t+m), d(t+m), Sn(t+m), B(t+m), h(t), D(t+m)\}$:

- t is the hour of the day.
- $T(t+m) \in \mathbb{R}$ is the temperature forecast for hour $t+m$.
- $d(t+m) \in \{0,1\}$ is a binary variable representing the type of day (workday or holiday).
- $Sn(t+m) \in \{0,1\}$ is a binary variable representing the type of season (hot or cold season).
- $B(t+m) \in \mathbb{R}_+$ is the predicted baseline loads for hour $t+m$.
- $h(t) \in \mathbb{R}$ stands for "history", and is a variable depicting the consumers' level of demand suppression resulting from previous responses. Consumers with high demand suppression are likely to have stronger "rebounds". Concretely it is defined as the discounted sum over the demand responses from previous hours:
$$h(t) = \frac{1}{N}\sum_{i=1}^{N}\sum_{\tau=0}^{t-1}(B_i(\tau) - L_i(\tau)) \cdot \sigma^{t-1-\tau}, \quad (1)$$
where $B_i(\tau)$ and $L_i(\tau)$ are the baseline and actual loads for consumer i, and $\sigma < 1$ is a discount factor.
- $D(t+m) \in \mathbb{R}$ is the target DR, i.e., how much the utility wants the average demand to change m hours later.

$a \in A$ is the action, which is the price signal for time $t+m$. We choose a discrete action space A which can represent the dynamic pricing programs with discrete pricing levels, as well as those with continuous prices after discretization.

$r \in \mathbb{R}_-$ is the reward which is defined as the negative squared error between the target and realized DR, i.e.,
$$r(t) = -(D(t) - (L(t) - B(t)))^2, \quad (2)$$
where $B(t)$ and $L(t)$ are the average baseline and realized loads of the *entire group of consumers*.

γ is a discount factor. We set it to be 1 as the horizon of the MDP, denoted by H, is of a relatively short length.

Importantly, we note that the above state definition is our *design choice* that captures important information about the "real" state in a compact way. As such, the formulated MDP is an approximation of the real-world process. We will demonstrate in our experiments that our model, albeit compact, is sufficient to achieve excellent performance.

III. Methodology: offline reinforcement learning

With the aforementioned MDP, our goal is to learn a pricing policy that maximizes the expected total rewards:
$$\pi^\star = \underset{\pi}{\mathrm{argmax}}(\mathbb{E}[\sum_{i=0}^{H} r(s_i, \pi(s_i))]). \quad (3)$$

To solve the problem in a safe and cost-efficient manner, we propose an *offline* RL framework: based on some *prior experiences* on PBDRP (which can be far from optimal), the utility aims to design a high-performance pricing policy *completely based on the existing data that is already collected by these prior experiences*, while never having to interact with the real world for this design task.

A. Utility Experience and Data Sets

As mentioned in Sec. I, many data sets available to utilities, such as the consumers' smart meter data under RTP programs, contain information about the consumers' responses to dynamic prices. These data sets can intuitively be used to design effective pricing policies. Generally, a data set with the following properties would enable effective offline learning: a) prices are dynamic (i.e., not flat), b) consumers' loads (and hence responses to prices) are measured, and c) the size of the data is sufficiently large to have reasonable coverage of diverse consumer behaviors. As we will show in our experiments, with data sets that satisfy these properties to just a reasonable extent, our offline learning methods will be highly effective in discovering high-performance new pricing policies.

B. Data Augmentation

A unique characteristic of DR programs is that the utility's goal is typically eliciting a *total* amount of DR from all the consumers. This implies that it is sufficient to focus on an aggregate consumer. However, focusing on an aggregate consumer also *reduces* the amount of data from which to learn, because the training data is all about aggregate consumption, and information for each consumer is not fully utilized.

To address this issue of reduced data set, we employ a simple but powerful data augmentation method: *bootstrapping* (i.e., re-sampling) from the full data set to create new data sets. In particular, instead of training on the full aggregation of size N, we can choose a smaller size $N' < N$, bootstrap a data set of size N' from the full data set of size N, and then aggregate the bootstrapped data. For example, with $N = 2,000$ consumers, the number of data sets of size $N' = 1,000$ would be $\binom{2000}{1000}$ — Collecting just a tiny fraction of these can already increase the total data size significantly. The reasoning behind this approach is that the aggregate behavior of $1,000$ random consumers would very closely indicate that of $2,000$ (modulo a scaling factor). In this way, we are able to greatly enrich the training data set so that the testing performance is further improved.

C. Offline Reinforcement Learning and Policy Evaluation

Offline RL and evaluation are inherently different from online ones in that it is doing counterfactual learning. One has to rely entirely on the past experience to answer "what if" questions. The distribution mismatch between the samples observed in the offline dataset collected by some behavior policy and that coming from the policy to be evaluated makes the offline learning very challenging.

One class of offline RL algorithms is to use dynamic program (DP) methods exploiting the mathematical structures of the Bellman equations to learn the counterfactuals. This is often referred to as "off-policy" learning. To be specific, one can learn the state or state-action value function via function approximation, and then derive the optimal policy from the value functions. The expected return can also be estimated: given a Q function [14] $Q_{\pi_d}(s,a)$ of some target policy π_d, and an initial state distribution S_0, the expected return can be calculated as follows:

$$J(\pi_d) = \mathbb{E}_{s_0 \sim S_0, a_0 \sim \pi_d(a_0|s_0)}[Q_{\pi_d}(s_0, a_0)]. \quad (4)$$

IV. Algorithm Design

A. Offline RL: Offline DQN with Extensions

We use Q-learning based algorithms as our offline RL algorithms [14]. Deep Q-network (DQN) [15] is a contemporary Q-learning algorithm which has been intensively studied and improved over the years. The algorithms we used in this paper are the offline variant of the DQN algorithms with extensions such as *double Q-learning* [16] and *dueling network* [17].

Algorithm 1: Offline DQN

```
Load the entire set of transitions $\mathcal{D} = \{(s, a, r, s')\}$ to
  replay memory;
Initialize Q network with random weights $\theta_0$;
for iteration 1 to $maxiter$ do
    Randomly sample mini-batch $\mathcal{D}' \subset \mathcal{D}$ ;
    Compute TD target for each tuple
      $(s_i, a_i, r_i, s'_i) \in \mathcal{D}'$:
          $y_i = r_i + \max_a Q(s'_i, a; \theta^-);$        (5)
    Perform stochastic gradient descent on
      $|y - Q(s, a; \theta)|^2$ w.r.t. $\theta$;
    For every $\mathcal{C}$ steps update $\theta^- = \theta$.
Output policy according to $Q_{\pi^\star} = Q(s, a; \theta)$
```

Algorithm 1 shows the vanilla offline DQN. The vanilla form is however observed to lead to overestimation of the TD target in certain applications. To alleviate this potential issue, the offline double DQN (DDQN) is introduced where separate parameters θ for action selection and θ^- for estimation are used for Bellman updates. To be specific, we substitute (5) in Algorithm 1 with the following equation:

$$y_i = r_i + Q(s'_i, \operatorname*{argmax}_a Q(s'_i, a; \theta); \theta^-); \quad (6)$$

Another extension involves the modification of the fully connected neural network (FCNN) architecture. Two parallel FCNNs sharing the same set of input features are a) trained to estimate the state value and advantage function separately, and b) combined by an aggregation layer to output the state-action value function estimates. Such a network is called the *dueling network*. We refer the reader to [17] for details.

B. Off-policy Policy Evaluation: Fitted Q-Evaluation (FQE)

Given an offline learned policy (or any policy of interest), one needs to demonstrate its performance before it can be implemented. Real-world pricing experiments are often costly to conduct and can have safety risks. A fully offline policy evaluation method is thus needed.

Interestingly, with a simple modification to algorithm 1, an off-policy policy evaluation (OPE) algorithm [18], [19] can be derived. To be specific, Q-learning-based RL algorithms learn the state-action values of the *optimal* policy and subsequently the *optimal* policy itself by choosing the actions that maximize the future state-action values. However, if the *max* operation in (5) is substituted with some external target policy π_d, the algorithm will learn the state-action values of the target policy $Q_{\pi_d}(s,a)$ instead of the optimal $Q^\star(s,a)$. This is the intuition behind fitted Q-Evaluation FQE [20]. To be specific, given a target policy π_d and an initial state distribution S_0, (5) is substituted with

$$y_i = r_i + Q(s'_i, a'_i = \pi_d(s'_i); \theta^-), \quad (7)$$

and the algorithm will output the target policy's expected return $J(\pi_d)$ according to (4) with $Q_{\pi_d} = Q(s, a; \theta)$.

As such, FQE serves as a tool to evaluate not only the pricing policies learned through offline DQNs, but also any other policies including heuristics.

V. Experiments

We conduct two sets of experiments to study the performance of the developed methods. First, we evaluate the offline RL algorithms based on an existing *real-world* data set where consumer responses to certain dynamic prices were recorded. Next, we further build a *simulation* environment that simulates consumer responses to prices. We then evaluate the algorithm performance using simulated data. Importantly, the simulation environment also enables us to perform *online* RL, something that cannot be performed with existing real-world data sets. As such, the online RL performance is evaluated, and it serves as a performance bound with which our offline RL algorithm performance can be compared.

A. Real World Test Case: Low Carbon London project

In this section, we demonstrate and evaluate the proposed algorithms based on a real-world data set. While there are existing RTP programs implemented by utilities, it is however difficult to access their data due to privacy and confidentiality constraints. In this work, we utilize an openly available data set from a research project led by UK Power Networks called *Low Carbon London* (LCL) [21], which successfully serves our purposes for evaluation.

1) Data collection and processing:

In the LCL project, smart meter data are collected from energy consumers who are subject to dynamic Time-of-Use (dToU) tariffs. The dToU tariff trial consists of three discrete pricing levels: high (£0.672/kWh), standard (£0.1176/kWh), and low (£0.0399/kWh). The prices were randomly scheduled and given to the participants one day ahead. A total of one year of consumer-level smart meter load data was collected in 2013, with 114 days being event days (i.e., during each of which at least one high or low price is scheduled). 1,122 consumers participated in the dTOU trial. In addition to the trial group, there was a reference group of 4,545 consumers receiving a flat rate tariff (£0.14228/kWh). All the consumers were located in the Greater London area. In addition to the tariff and smart meter data, we also collected the temperature of London in 2013 [22]. The LCL study also provides a linear regression model for computing the *baseline* loads.

Based on the LCL data set, we implemented the bootstrapping technique introduced in Sec. III-B. $\frac{1}{5}$ of the consumers are randomly sampled (with replacement) each time. A total of 400 re-sampling are performed to generate approximately $100K$ hours of offline experience. We then follow the steps in Sec. II-A to construct the MDP experience.

Advanced notice and delayed rewards: While the dToU program from the LCL project announces the event/dynamic prices one day ahead, we made the following observations from analyzing the data: Despite the DA notification, consumers tend to still follow their normal consumption pattern up until 3 to 4 hours ahead of the scheduled event (with either

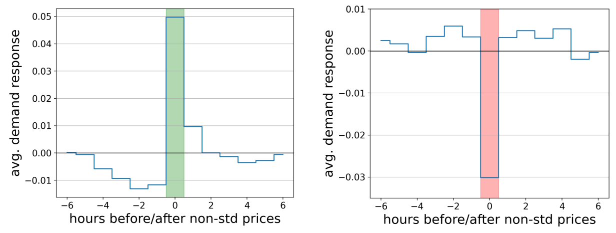


Fig. 1. Demand response before and after the low (green) and high (red) price events, averaged over all consumers and all such events, in the *Low Carbon London* dToU trial.

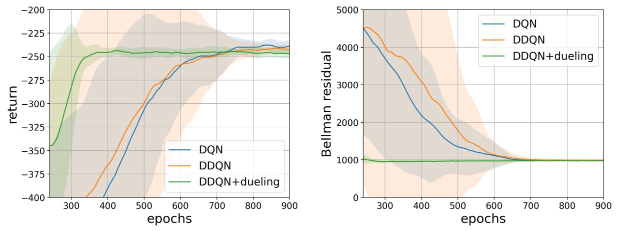


Fig. 2. Offline RL learning curves: Q value function return (left) and Bellman residual (right).

a high or low price), as shown in Fig. 1. In other words, we can accurately approximate the day-ahead program by a *3-hour* ahead pricing program (cf. Sec. II with $m = 3$).

2) Offline learning and evaluation results:

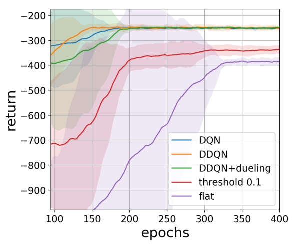
The offline DQN learning algorithms (vanilla DQN, DDQN, and DDQN with dueling networks) are evaluated on the above constructed data set based on LCL's real-world data. The architectures of DQN and DDQN models are identical: a FCNN with an input layer that takes in 8 state features, a hidden layer with 18 neurons, and an output layer that produces 3 outputs, each corresponding to a value of the Q function with one (out of 3) possible actions (i.e., high, standard, and low prices). For the dueling network, each one of the two FCNNs has a single hidden layer with 18 neurons.

Fig. 2 shows the policy returns and the Bellman residual learning curves. The mean (solid lines) and standard deviation (shadowed intervals) are plotted based on 10 repeated runs. The returns of 3 offline RL algorithms all converged at around -245, indicating that *their performance are very close to each other* albeit with different convergence speeds.

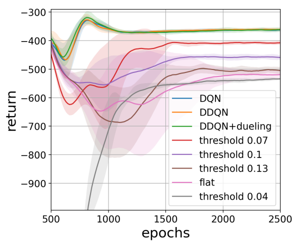
Next, the offline-learned policies are evaluated using offline FQE along with some heuristics including

- flat rate with standard price only, and
- a simple and intuitive threshold-based policy: high price when target $D > 0.1$, low price when $D < -0.1$, and standard price otherwise.

Offline FQE provides a way to compare and rank all these policies. We plot the FQE results in Fig. 3(a). It is clear that the flat rate policy is the worst with no demand response control ability at all, followed by simple threshold-based policy which is far from optimal. The best policies are the ones learned using offline RL, with the converged returns all at around -245. Notably, this evaluation (-245) is consistent with the above converged offline RL returns (-245).



(a) LCL test case



(b) 3HA3p simulation environment

Fig. 3. Offline FQE returns.

B. *Simulation Environments: Comparing the Proposed Offline Algorithms to Online Benchmarks*

1) Motivations: In this section, we build several simulation environments to model the price responses of consumers in a variety of dynamic pricing programs. With the simulation environments, the actual pricing actions and responses can be simulated indefinitely, and this enables very accurate performance evaluation of any policy, including the ones learned using offline RL methods. Importantly, the simulation environments also serve as testbeds to perform *online* RL which would not be possible with existing real-world data sets. As such, by implementing the state of the art online RL algorithms, their performance provide a performance bound with which offline RL policies can be compared. The simulation environments also allow us to verify the efficacy of offline policy evaluation methods.

2) Establishing the simulation environments: We utilize the idea of *price elasticity matrix* [23] to model consumer response to dynamic pricing. At any given time, the load response of the consumer is affected by the entire history of price signals they observed. We use c_{it} to quantitatively represent the effect of price at time i on the load at time t. Concretely, we define the load-price coefficient:

$$c_{i,t} = \frac{L_{i,t} - B_{i,t}}{p(i) - p_n},$$

where $p(i)$ is the price at time i, p_n is the normal (i.e., standard) price, and L_{it} and B_{it} are the components of load consumption and baseline estimation at time t affected by price at time i. $\sum_i L_{it} = L(t)$ and $\sum_i B_{it} = B(t)$. The load response of consumer at time t can then be written as the summation of load response components over time:

$$L(t) - B(t) = \sum_{i=1}^{t+m} c_{i,t} \left(p(i) - p_n\right) \tag{8}$$

Note that the load response at time t is affected by prices from hour 1 to hour $t+m$, i.e., all the price signals that the consumers can observe at time t. The values of the coefficients are set so that the simulation environment mimics the consumer behaviors observed in the LCL dataset.

The environment is designed in the following way:

- An episode has a length of 24-m hours.
- At hour 1, the state is randomly initialized. The target curves are randomly generated.
- At each time step t, given an action a_t that corresponds to the price to be realized at time $t+m$, the load response is calculated following (8), and the history feature is calculated following (1). The state is then updated as in Sec. II-A, and the reward is calculated as in (2).
- At hour $24-m$, the final price signal a_{24-m} (i.e., the price for hour 24) is announced, and the terminal reward is calculated based on the remaining DR actions in the final hours.

We choose $m = 1$ and 3 to represent the hour-ahead (HA) and day-ahead (DA) programs (cf. Section V-A1), respectively. We also use two action spaces $|A| = 3$ and $|A| = 7$ to model both dToU-style RTP programs with discrete price settings, and the continuous price RTP programs by discretizing the prices into 7 levels. In total 4 environments are designed, namely the 1-hour-ahead program with 3 discrete prices (1HA3p), 1-hour-ahead program with 7 discrete prices (1HA7p), 3-hour-ahead program with 3 discrete prices (3HA3p), and 3-hour-ahead pricing with 7 discrete prices (3HA7p).

3) Offline data collection and learning: We collect one year of load data through consumer interactions with the environments using uniformly distributed random prices as the behavior policy. The external variables independent of the price actions such as time, weather and baseline consumption are generated based on the LCL datasets. The datasets are then augmented using random targets. Approximately $300K$ data (i.e., state transitions) are collected. The same set of offline RL algorithms as above are implemented to train the policies.

4) Offline and online policy evaluation: Enabled by the simulation environment, we first demonstrate the efficacy of our offline evaluation method, i.e., FQE. In particular, as actual rewards can be observed with the simulation environment, we can simulate any policy and easily compute the "ground truth" average rewards with which the FQE results can be compared. Here, we focus on the 3HA3p case which corresponds to the LCL case studied in Section V-A. The heuristics that we evaluate again include flat price and simple threshold-based policies. In particular, we evaluate the threshold-based policies with 4 different thresholds, ±0.04, ±0.07, ±0.1, and ±0.13.

TABLE. I. OFFLINE AND ONLINE EVALUATION (3HA3P).

	offline FQE	sim eval
DQN	-355	-393
DDQN	-351	-394
DDQN+dueling	-353	-395
threshold 0.07	-392	-452
threshold 0.1	-453	-509
threshold 0.13	-491	-594
flat	-505	-620
threshold 0.04	-526	-674

Fig. 3(b) and Table I show the evaluation results using offline FQE and online simulation evaluation. We observe that offline FQE is a sufficiently accurate indicator of the accurate online-evaluated policy performance: a) FQE provides rela-

TABLE. II. PERFORMANCE COMPARISON FOR OFFLINE LEARNED POLICIES, ONLINE BENCHMARKS, AND HEURISTICS.

simulation setting	offline DQN	offline DDQN	offline DDQN+ dueling	online DQN	online DDQN	online DDQN+ dueling	online PPO	threshold	flat
1HA3p	-422	-424	-422	-418	-417	-416	-417	-486	-672
1HA7p	-220	-220	-224	-214	-214	-215	-229	-302	-669
3HA3p	-393	-394	-395	-392	-392	-391	-386	-452	-620
3HA7p	-194	-194	-196	-183	-183	-182	-188	-355	-646

tively similar average reward estimates, and more importantly, b) highly accurate ranking of the policies.

5) Online learning benchmarks:

Last but not least, we employ several state-of-the-art online RL algorithms as performance benchmarks with which our offline methods are compared. These include value-based ones such as online vanilla DQN, online DDQN, online DDQN with dueling networks, and actor-critic methods such as the Proximal Policy Optimization (PPO) with clipping [24].

Table II summarizes the performance of offline learned policies, online benchmarks, and simple heuristics in numeric values. We observe that the offline learned policies not only a) outperform heuristic policies by large margins, but are also b) very close to the online benchmarks. These demonstrate great efficacy of the developed offline RL methods and provide assurance of the high performance of the offline learned pricing policies for DR.

VI. CONCLUSION

We have developed fully offline RL methods to design dynamic pricing DR programs based on existing data sets without any further interactions with consumers. We have formulated the dynamic pricing design problem in a sequential decision-making framework with an MDP. We have developed a novel data augmentation method that greatly increases the size of the data set. We have developed DQN-based offline RL and policy evaluation methods to learn high-performance pricing policies. We have extensively evaluated the developed offline learning methods with both a real-world data set and simulation environments. Excellent performance of the proposed offline RL method is demonstrated which approaches the ideal performance bound provided by state-of-the-art online RL algorithms.

REFERENCES

[1] R. Mieth and Y. Dvorkin, "Online learning for network constrained demand response pricing in distribution systems," *IEEE Transactions on Smart Grid*, vol. 11, no. 3, pp. 2563–2575, 2019.

[2] Z. Wen, D. O'Neill, and H. Maei, "Optimal demand response using device-based reinforcement learning," *IEEE Trans. on Smart Grid*, vol. 6, no. 5, pp. 2312–2324, 2015.

[3] Y. Li, Q. Hu, and N. Li, "A reliability-aware multi-armed bandit approach to learn and select users in demand response," *Automatica*, vol. 119, p. 109015, 2020.

[4] X. Kong, D. Kong, J. Yao, L. Bai, and J. Xiao, "Online pricing of demand response based on long short-term memory and reinforcement learning," *Applied energy*, vol. 271, p. 114945, 2020.

[5] A. Shojaeighadikolaei, A. Ghasemi, K. R. Jones, A. G. Bardas, M. Hashemi, and R. Ahmadi, "Demand responsive dynamic pricing framework for prosumer dominated microgrids using multiagent reinforcement learning," in *2020 52nd North American Power Symposium*. IEEE, 2021, pp. 1–6.

[6] G. Barbose, C. Goldman, and B. Neenan, "A survey of utility experience with real time pricing," Lawrence Berkeley National Lab, Berkeley, CA, Tech. Rep., 2004.

[7] B.-G. Kim, Y. Zhang, M. Van Der Schaar, and J.-W. Lee, "Dynamic pricing and energy consumption scheduling with reinforcement learning," *IEEE Trans. on smart grid*, vol. 7, no. 5, pp. 2187–2198, 2015.

[8] R. Lu, S. H. Hong, and X. Zhang, "A dynamic pricing demand response algorithm for smart grid: reinforcement learning approach," *Applied Energy*, vol. 220, pp. 220–230, 2018.

[9] A. Ghasemkhani and L. Yang, "Reinforcement learning based pricing for demand response," in *IEEE Int'l Conf. on Communications Workshops*. IEEE, 2018, pp. 1–6.

[10] N. Nezamoddini and Y. Wang, "Real-time electricity pricing for industrial customers: Survey and case studies in the united states," *Applied energy*, vol. 195, pp. 1023–1037, 2017.

[11] K.-C. Lee, H.-T. Yang, and W. Tang, "Data-driven online interactive bidding strategy for demand response," *Applied Energy*, vol. 319, p. 119082, 2022.

[12] F. Ruelens, B. J. Claessens, S. Quaiyum, B. De Schutter, R. Babuška, and R. Belmans, "Reinforcement learning applied to an electric water heater: from theory to practice," *IEEE Trans. on Smart Grid*, vol. 9, no. 4, pp. 3792–3800, 2016.

[13] D. Jang, L. Spangher, T. Srivistava, M. Khattar, U. Agwan, S. Nadarajah, and C. Spanos, "Offline-online reinforcement learning for energy pricing in office demand response: lowering energy and data costs," in *Proc. of the 8th ACM Int'l Conf. on Systems for Energy-Efficient Buildings, Cities, and Transportation*, 2021, pp. 131–139.

[14] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[16] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proc. of the AAAI conf. on artificial intelligence*, vol. 30, no. 1, 2016.

[17] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *Int'l Conf. on machine learning*, 2016, pp. 1995–2003.

[18] D. Lyu, B. Liu, M. Geist, W. Dong, S. Biaz, and Q. Wang, "Stable and efficient policy evaluation," *IEEE transactions on neural networks and learning systems*, 2018.

[19] B. Liu, J. Liu, M. Ghavamzadeh, S. Mahadevan, and M. Petrik, "Proximal gradient temporal difference learning algorithms," in *International Joint Conference of Artificial Intelligence (IJCAI)*, 2016, pp. 4195–4199.

[20] H. Le, C. Voloshin, and Y. Yue, "Batch policy learning under constraints," in *Int'l Conf. on Machine Learning*, 2019, pp. 3703–3712.

[21] J. R. Schofield, R. Carmichael, S. Tindemans, M. Bilton, M. Woolf, G. Strbac *et al.*, "Low carbon london project: Data from the dynamic time-of-use electricity pricing trial, 2013," *UK Data Service, SN*, vol. 7857, no. 2015, pp. 7857–7851, 2015.

[22] "Met office MIDAS open: UK land surface stations data (1853-current)," Centre for Environmental Data Analysis, 2019.

[23] X. Qu, H. Hui, S. Yang, Y. Li, and Y. Ding, "Price elasticity matrix of demand in power system considering demand response programs," in *IOP Conf. Series: Earth and Environmental Science*, vol. 121, no. 5. IOP Publishing, 2018.

[24] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.