Mitigation of Coincident Peak Charges via Approximate Dynamic Programming

Chase P. Dowling and Baosen Zhang

***Abstract*— A significant portion of a consumer's annual electrical costs can be made up of coincident peak charges: a transmission surcharge for power consumed when the entire system is at peak demand. This charge occurs only a few times annually, but with per-MW prices orders of magnitudes higher than non-peak times. While predicting the moment of peak demand charges over the course of the entire billing period is possible, optimal cost mitigation strategies based on these predictions have not been explored. In this paper we cast coincident peak cost mitigation as an optimization problem and analyze conditions for optimal and near-optimal policies for mitigation. For small consumers we use approximate dynamic programming to first show the existence of a near-optimal policy and second train a neural policy for curtailing coincident peak charges when subject to ramping constraints.**

I. Introduction

A coincident peak (CP) is a consumer's electrical demand at the time of the total system peak demand. Since much of the power system infrastructure is only used only during peak times [1], some system operators and utilities use CP pricing mechanisms to incentivize customers to reduce their consumption during peak times, therefore hoping to achieve an overall reduction of the system peak [2], [3]. Existing CP charges are applied through a rate structure, with the rates at peak times hundreds of times larger than at regular times. As a result, CP charges often account for a significant portion—often greater than 20%—of annual electrical costs for participating customers [4], providing them with a strong incentive to reduce their consumption at these peak times [5], [6].

In this paper, we adopt the view point of a small customer facing CP charges and study how the customer can operationally mitigate this cost. The primary challenge is that the timing of the CP charges are only known after all of the system demands have been realized. For example, if CP is charged on a monthly basis [3], the hour that the peak load occurred in only determined after the entire month has passed.

To mitigate this uncertainty in peak timing, operators typically provide warning signals to consumers to indicate peak is forthcoming. In [4], the authors utilize these signals to develop a scheduling model for a data center's workload in the Fort Collins PUD [3]. However, forecasting when a peak will occur is a difficult prediction problem [7], [8], since it only occurs (by definition) at a single point in time. Since the rate associated with the CP is orders of magnitude higher than normal time-of-use rates, false negative predictions are extremely costly. Therefore operators tend to send out many successive CP warning signals, degrading the efficiency of customer responses and leading to user fatigue in the long run [9], [6].

In this work we treat the problem of mitigating CP costs as an optimization problem that is continually solved over the entire horizon of the billing period. Instead of explicitly predicting when the peak will occur, we adopt a probabilistic framework to gracefully incorporate observations made by the customer to maximize their expected revenue. That is, at each time-step we calculate the probability of the peak occurring at some point in the future having observed previous values of system demand.

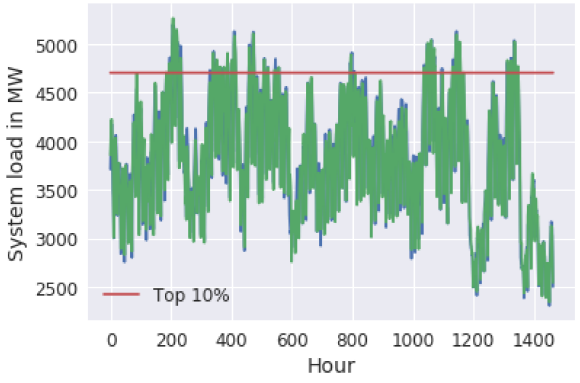
Related works on mitigating CP pricing focus on large consumers with considerable demand flexibility, namely, data centers [10]. Limited works have addressed CP prices for data center consumers directly such as [4] which incorporates existing grid operator signals. Others related to data center peak power consumption address the problem generally based on time-of-use costs given on-site storage or generation capabilities [11], [12] without tackling the idiosyncrasies of CP pricing mechanisms.

Dynamic programming is a natural approach to maximize the expected revenue of a small customer in the face of CP timing uncertainty. However, since the action space of a customer is continuous and coupled in time, solving the dynamical programming problem becomes intractable. Therefore we approximate the value function and train a deterministic policy parametrized as a neural network. Based on the structure of the CP charge, we design the input of the neural network to explicitly include the maximum of the observed demand and the number of time periods. Using these inputs, we show that this neural network based policy is comparable to a brute force grid search and outperforms a standard benchmark algorithm. This approach advances the state-of-the-art by providing a way to actively reduce the CP cost that does not rely on system warning signals or assumes an adversarial environment.

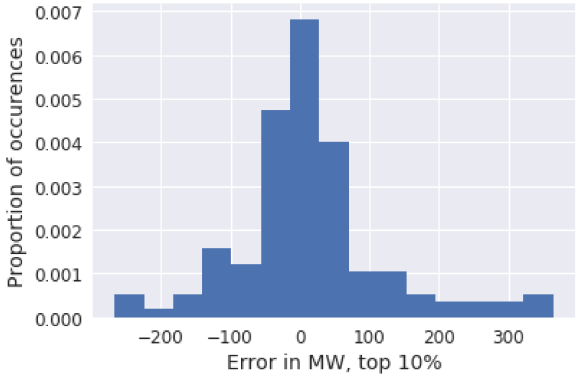
The rest of the paper is organized as follows: Section II defines the optimization problems to be solved, Section III provides the solution framework, Section IV presents a numerical case study. We conclude with a discussion on future work for both large and small consumers and make some final remarks in Section V.

Department of Electrical and Computer Engineering, University of Washington Seattle, WA 98195, USA {cdowling, zhangbao}@uw.edu

This work is partially supported by grants from Centrica, plc.



(a)



(b)

Fig. 1: In PJM's Duke Energy Ohio/Kentucky region: (a) actual and forecasted system load and (b) the distribution of forecast error from June 1 to September 30, 2018.

II. Model and Problem Formulation

We consider a small customer that tries to maximize its revenue subject to CP charges over T time periods. Let x_t be the energy consumption of the customer at time $t \in \{1, \dots, T\}$, and it is limited to be between $\underline{x}$ and $\bar{x}$. We assume the revenue of the customer is represented by a concave increasing function $g(\cdot)$ [13], [14] of power consumption. Let π_{cp} be the CP charge rate and the customer pays an amount of $\pi_{cp} x_{t^*}$ where t^* is the time period the system peak occurs.

We model the system load in each time period as random variables $S_1, \dots, S_T$ where the mean of S_t is the forecasted load value. Even though system loads are strongly correlated in time, once the forecast value is given, the forecast errors are typically independent across time periods [15], [16]. For example, Fig. 1 illustrates the distribution of system load forecast error for the top 10% of system load values during summer months in 2018 in a single PJM subregion.

Additionally, it is clear a large customer has a measurable impact on the value of S_t. In the case of ERCOT, for example, the 6 largest customers accounted for over 80% of each summer monthly CP in 2017 and 2018. On the other hand, of the 130 total customers participating in ERCOT's CP pricing program, 20% in 2017, and 40% in 2018 consumed less than $\frac{1}{10}$ the difference between the annual system peak and the second closest system demand. For these exceedingly small customers, the variance of the forecast error well exceeds their CP demands of less than 5-10 MW.

Therefore, in the case of a small customer we assume that $S_1, \dots, S_T$ are independent and their mean is given by the forecast values. We define t^* as the time index corresponding to the maximum load:

$$t^* = \arg\max_t \{S_1, \dots, S_t\}. \tag{1}$$

Note since t^* is a function of random variables, it is also a (discrete) random variable.

With these definitions, the expected net revenue (or reward) of a customer is

$$\mathbb{E}[R] := \mathbb{E}\left[\sum_{t=1}^{T} g(x_t) - \pi_{cp} x_{t^*}\right] \tag{2}$$

where t^* is defined in (1) and the expectation is over the random variables $S_1, \dots, S_T$.

The goal of the customer is then to maximize $\mathbb{E}[R]$ subject to their operational constraints. We assume a fairly simple customer model where the demand from each time-period is coupled through a ramping constraint, and the customer's optimization problem is

$$\begin{aligned} \underset{x_t}{\text{maximize}} \quad & \mathbb{E}[R] \\ \text{subject to} \quad & \underline{x} \le x_t \le \bar{x} \\ & x_{t-1} - \delta \le x_t \le x_{t-1} + \delta \end{aligned} \tag{3}$$

where δ limits the possible rate-of-change between two time periods. Other types of constraints can be included using the techniques described in this paper.

A. Sequential Decision Problem

In practice, the optimization problem in (3) needs to be solved in a sequential manner. There are two sources of temporal coupling in (3) that makes this sequential optimization problem nontrivial and interesting. The first is the ramp constraint between two successive time steps. The second is how the timing of the peak changes after loads are observed.

At time t, the customer have observed the realization of $S_1, \dots, S_{t-1}$, which we denote as $s_1, \dots, s_{t-1}$. Based on these observations, the value of peak time t^* changes. In other words, if the observed loads are large, then the peak is likely to have already occurred and the customer can act aggressively; conversely, if the maximum value of the observed loads are small, then the customer should act more conservatively to protect against incurring a large CP charge in the future. Therefore, even if the ramp limits are not present, the structure of the CP charge induces a time dependency on the observations made at each stage. A variant of (3) without ramp constraints is studied in [4] where data centers are assumed to have a large amount of flexibility and can ignore the coupling of its own actions between two time periods. In this paper, we focus on commercial customers that may not have this level of flexibility and are limited in their rate-of-change.

B. Benchmark Algorithm

There are two typical strategies to solve (3) in practice. The first is to simply assume that all time periods (e.g., all hours between 3 PM and 7 PM on a hot summer day) experience the peak demand and conservatively reduce the load to mitigate the CP charge [6]. The the CP charge is evenly distributed over all of these time intervals. The second is to follow the warning signals of operators and treat those as true peak times [3], [4]. It turns out that these two strategies amount to the same thing, since operators tend to be conservative and issue CP warnings for all of the time periods that have a reasonable chance of experiencing the moment of peak demand [3]. This is to say that conservative CP warnings amount to treating any hot summer afternoon, for example, as equally likely to the system peak without taking into joint consideration the known system capacity and previously observed system loads during the billing window. Therefore we adopt the following strategy as the a baseline algorithm which we call the naive strategy [8], where the customer solves

$$\max_{x_t} \; g(x_t) - \frac{1}{T}\pi_{cp} x_t, \tag{4}$$

where the scaling factor $1/T$ represents the fact that the cost of CP is amortized evenly to all of the time periods under consideration. The optimal solution is then the the demand that satisfies the first order optimality condition $Tg'(x^*) - \pi_{cp} = 0$.

Note even though this solution is simple to compute, it does not take into account the successive realization in the system load and is generally suboptimal. In later comparisons, we will call it the naive policy. In the next section, we develop a policy based on approximate dynamic programming to solve (3).

III. Approximate Dynamic Programming

A. Dynamic Programming Formulation

Let us first directly apply a dynamic programming approach to optimize (2). Suppose the customer is solving for the optimal x_T, having already chosen $x_1, \ldots, x_{T-1}$ at the final step $t = T-1$. The customer must maximize the expected reward conditioned on observed system load realizations, $s_1, \ldots, s_{T-1}$, specifically, $\mathbb{E}[R|s_1, \ldots, s_{T-1}]$.

At $t = T-1$, let $s_m = \max\{s_1, \ldots, s_{T-1}\}$, the maximum observed so far; since this is the final round, the expected reward depends *only* on whether s_T will be larger than s_m. Let $p_T = 1 - P(s_m < S_T)$, the probability that the final system load realization will be the CP. Then the objective,

$$\mathbb{E}[R|s_m] = \sum_{t=1,\ldots T-1} g(x_t) + g(x_T) - \pi_{cp}\mathbb{E}\left[x_{t^*}|s_m\right] \tag{5a}$$

$$= \sum_{t=1,\ldots T-1} g(x_t) + g(x_T) - \tag{5b}$$

$$\pi_{cp}[(1-p_T)x_{t^*} + p_T x_T]. \tag{5c}$$

Thus, for the solution x'_T to $g(x'_T) - \pi_{cp} p_T = 0$, the optimal x^*_T is the point in the interval $[x_{T-1} - \delta, x_{T-1} + \delta]$ which minimizes $|x'_T - x^*_T|$. At $t = T-2$, in order to solve for the optimal x^*_{T-1} there are two potential rounds that the CP may yet occur on and the customer must consider the probability that either S_T or S_{T-1} is the CP. Indeed,

$$\mathbb{E}[R|s_1, \ldots, s_{T-2}] = \sum_{t=1,\ldots,T-2} g(x_t) + g(x_{T-1}) + \tag{6a}$$

$$\mathbb{E}[g(x_T) - \pi_{cp}[(1-p_T)x_{t^*} + p_T x_T]], \tag{6b}$$

noting that x_T remains inside the expectation since it depends on the realization of S_{T-1}. Iterating backwards yields a dependency on future realizations of S_t, where only the current consumption x_t, maximum system load observed thus far s_m, and number of rounds remaining $T-t$ influence future choices of $x_{t+1}, \ldots, x_T$.

A straightforward means of addressing this would be a brute force grid search. Consumption values in $[\underline{x}, \bar{x}]$ and a range of likely system loads S_t can be discretized, with every potential outcome being computed forward from each possible initial value x_1. At each time a consumer would choose a feasible x_{t+1} subject to ramping constraints that maximizes the expected reward over the *entire* horizon T for all possible outcomes given $s_1, \ldots, s_t$ using the output of the grid search as a look-up table; however, a complete grid search exhibits exponential complexity in T. This dimensionality problem is common in applications of dynamic programming [17].

Therefore we propose an approximate dynamic programming approach by sampling from all possible outcomes in order to estimate the best choices of x_{t+1}. These samples are used to train a policy f, which takes as input at time t the current consumption x_t, the largest system load observed so far in the billing period s_m, and the number of rounds left, $T-t$. The policy then outputs an estimated optimal $\hat{x}_{t+1}$. We note that in the absence of these time coupling constraints, an optimal solution exists since each time-step is completely independent.

B. Neural Network Policy

Neural networks have gained popularity as a tractable way to parameterize policies. For example, they have been used to solve approximate dynamic programming problems in [17], [18], [19]. We also adopt a neural network based policy to solve (3). In the context of dynamic programming, a policy is a function that maps previous values to an action. In our case, this policy should map the current choice of x_t and observations of s_t to an output x_{t+1} that a customer should select as their demand based on, in this case, criterion that maximizes their expected reward over the remaining time horizon. A policy in the context of approximate dynamic programming attempts to output a value $\hat{x}_{t+1}$ that is close to optimal.

Therefore, in order to train a policy f we require an approximation of the true optimal output x^*_{t+1} of f that maximizes expected reward R given previous observations of s. Alg. 1 details the process by which these samples are generated. At time t, for each feasible value of x_{t+1}

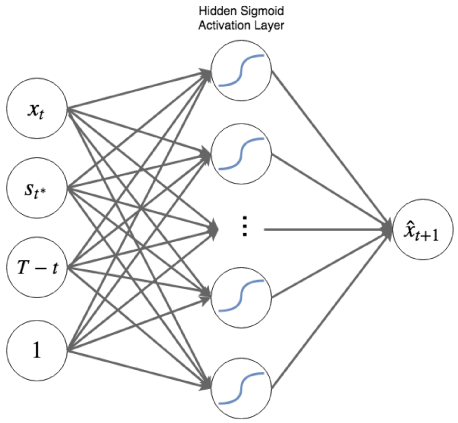


Fig. 2: Architecture of single-layer neural network policy, with inputs x_t, s_m, $T-t$, and a linear bias term.

subject to the ramping constraints, potential outcomes are forward simulated until time T a total of C times. The value of x_{t+1} with the best average *remaining* reward—modulo $\sum_{i=1}^{t} g(x_t)$—is selected as the training output. If the range of customer consumption values are discretized into n values, sampling across all possible starting times $t \in T$ yields an improved complexity of $\mathcal{O}(TCn)$.

Many samples are compiled and used to train a neural network to to approximate the function f. Fig. 2 illustrates the basic network architecture described and used in Sec. IV. The necessary inputs of the policy at time t are 1) the current state x_t, as this determines the range of feasible values due to the ramping constraint, 2) the maximum value $s_m = \max\{s_1, \ldots, s_t\}$ observed thus far, as values of $s_i < s_m$ have no bearing on the timing of the CP, and 3) the number of rounds left, $T-t$—given the probability density function of S_t—determines the probability any future value will be greater than s_m and thus the new potential CP. When performing grid search, these three values completely determine the expected reward when choosing a value x_{t+1}.

IV. Case Studies

To test the efficacy of our approximate dynamic programming solution compared to the naive strategy, we set up a numerical study[1]. We consider two different consumer revenue functions (illustrated in Fig. 3),

$$g_1(x) = 2\log(1+x^2) \text{ and} \tag{7a}$$
$$g_2(x) = 1.386\sqrt[4]{x} \tag{7b}$$

For both revenue functions we suppose the customer's ramp constraint $\delta = 0.3$, and that the customer's CP charge rate $\pi_{cp} = 0.6Tg(\bar{x})$, or 60% of their maximum possible gross revenue over T rounds. Typically CP charges form greater than 20% of their annual electrical costs, but we choose to much higher percentage to illustrate a more drastic scenario.

[1]All code to reproduce our results and results plots can be found on our GitHub project repo at `https://github.com/cpatdowling/peakload/blob/master/notebooks/cdc_2019_submission.ipynb`

Data: x_t, s_m, T
Result: Estimated policy output $\hat{x}_{t+1}$
$C =$ number of Monte Carlo simulations;
sim_rewards $= []$;
discretize $[x_t - \delta, x_t + \delta]$;
for *each feasible* $x_{t+1} \in [x_t - \delta, x_t + \delta]$ **do**
  sim_rewards$[x_{t+1}] \leftarrow []$;
  for $j = 1, \ldots, C$ **do**
    sample s_{t+1} according to system load distribution;
    $\mathbf{x} \leftarrow [x_t, x_{t+1}]$;
    $\mathbf{s} \leftarrow [s_m, s_{t+1}]$;
    for $k = t+2, \ldots, T$ **do**
      sample s_k according to system load distribution;
      $\mathbf{s} \leftarrow s_k$;
      randomly sample feasible x_k from interval $[x_{k-1} - \delta, x_{k-1} + \delta]$;
      $\mathbf{x} \leftarrow x_k$
    end
    sim_rewards$[x_{t+1}] \leftarrow R(\mathbf{x}, \mathbf{s})$;
  end
end
$\hat{x}_{t+1} = \arg\max_{x_{t+1}} \frac{1}{C} \sum$ sim_rewards (choose x_{t+1} with best average forward simulated reward)

Algorithm 1: Monte Carlo path sampling

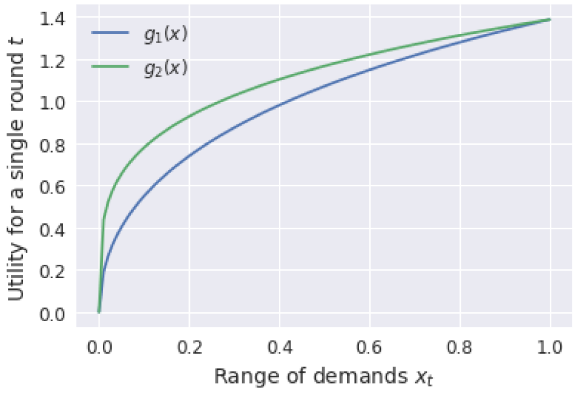


Fig. 3: Case study revenue functions $g_i(x)$

For $T = 2, \ldots 10$ rounds[2], we use the sampling strategy defined in Alg. 1 to generate 1000 input/output samples per time $t \in [1, \ldots, T]$, such that we train with an even number of $\hat{x}_{t+1}$ for all t. For each feasible x_{t+1} being evaluated, the number of simulations $C = 100$.

As a first pass we design our neural network to have a single hidden layer of size 4 with a sigmoidal activation function. Further, we included a linear bias term, indicated as the fourth input in Fig. 2. The neural network is trained using mean-squared error; additional hyperparameters like learning

[2]Results for larger values of T can be found for $g_1(x)$ in our Github project repo

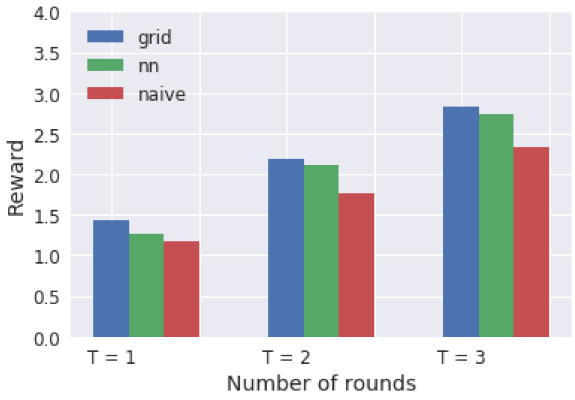


Fig. 4: Comparison of best-possible performance via grid search to a NN policy and the naive strategy. Reward is strictly increasing with each additional number of rounds roughly as $Tg_1(x)$

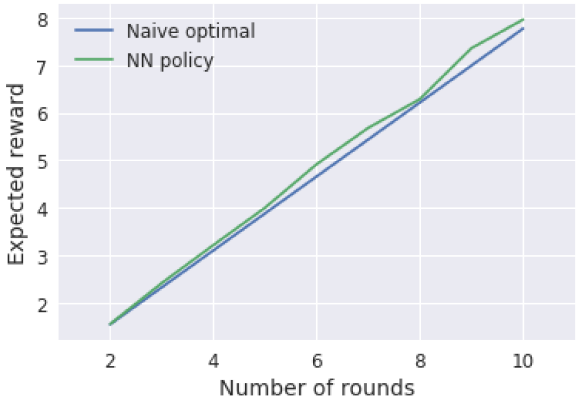


Fig. 6: Policy performance for revenue $g_2(x)$, across time horizons T with π_{cp} set to be 60% of maximum, unpenalized revenue for each T.

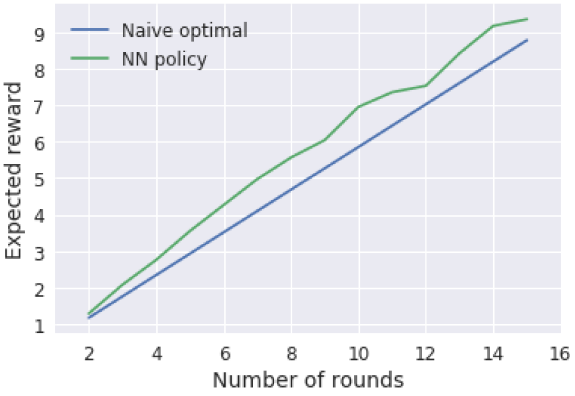


Fig. 5: Policy performance for revenue $g_1(x)$, across time horizons T with π_{cp} set to be 60% of maximum, unpenalized revenue for each T.

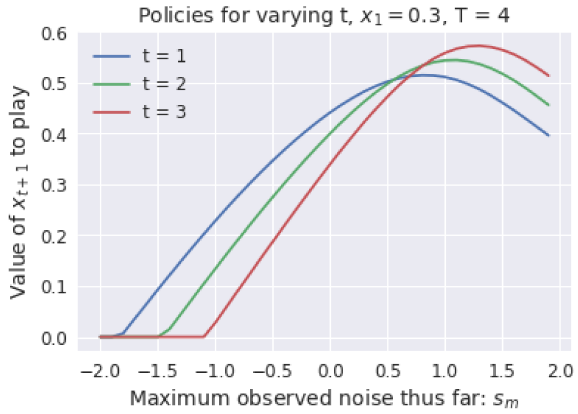


Fig. 7: Example of policy for revenue $g_1(x)$ for multiple rounds t over time horizon $T = 4$ and fixed $x_t = 0.3$. With later rounds of t, the policy becomes less conservative and shifts to the right as the decreasing number of rounds decreases the probability of a new maximum system load being observed.

rate and batch-size can be found in our linked repository.

First we test the validity of the assumption that our ADP sampling procedure yields near-optimal choices $\hat{x}_{t+1}$ against an exhaustive grid-search. For $T = 2, 3$ and 4, and revenue function $g_1(x)$ we compute an exhaustive grid for our example function and system load distribution. Fig. 4 illustrates the relative performances of each strategy; while we found that the discretization resolution of the grid search has a noticeable effect on the resulting reward given the choice of our case study revenue function, the NN policy performs nearly as well and we make use of the sampling technique on a larger number of rounds for which grid search is intractable.

Figures 5 and 6 illustrate the performance of the respective NN policies against the naive strategies for $g_1(x)$ ad $g_2(x)$. The NN policies consistently outperforms the naive optimal solution while maintaining the added benefit of being an solution approximated from sampled paths. In the case of mitigating CP costs on an hourly basis, it is unlikely that a potential CP would occur at anytime in excess of 8 to 10 consecutive hours, viz occuring outside known, afternoon peak hours [4]. This benefits a customer by allowing them to focus on sweeping training parameters to tune the policy for a narrow range of values of T.

Figures 7 and 8 illustrate examples of the outputs of f corresponding to $g_1(x)$ and $g_2(x)$ both trained for $T = 4$. For each time $t = 1, 2, 3$, the output x_{t+1} is given as a function of s_m for a fixed initial value, $x_t = 0.3$. Note that with each consecutive round, the likelihood of S_t remaining that may be a CP changes both as a function t *and* s_m. In general, the probability that the next realization of S_t will be the maximum over all T, $p_t = \frac{1}{T-t}(1 - P(S_t < s_m)^{(T-t)}$. Interestingly these policies learned for $g_1(x)$ and $g_2(x)$ appear very different.

In the case of policy f learned for $g_1(x)$, outputs of the policy for each t become *less* conservative with decreasing number of rounds. The flattening of the policy at small

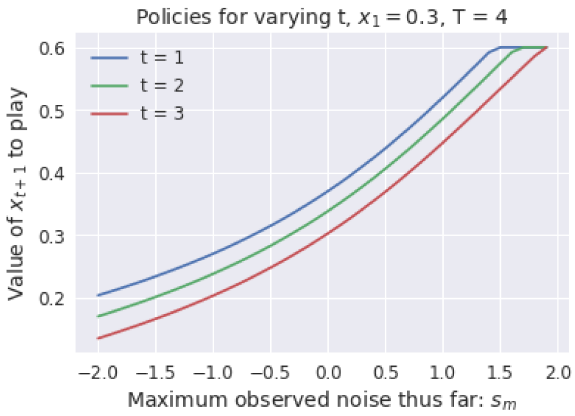


Fig. 8: Example of policy for revenue $g_2(x)$ for multiple rounds t over time horizon $T = 4$ and fixed $x_t = 0.3$. With later rounds of t, the policy interestingly becomes *more* conservative, likely due to the sharper decrease in $g_2'(x)$ in increasing x.

values of s_m is due to the ramping constraint. Conversely, decreasing the radius of curvature of the revenue function—the sharpness in the initial revenue increase transitioning into diminishing returns—as in $g_2(x)$, it appears that a *more* conservative strategy arises, likely due to the decreased cost of false negatives. Large changes in x result in little change in $g_2(x)$ but correspondingly a relatively larger marginal decrease of π_{cp} in the CP charge. That is to say it costs the customer little to curtail to values already near the naive optimal strategy for $g_2(x)$, (e.g. for $T = 10$, $x = 0.311$), yet the NN policy still improves on the naive strategy.

V. Conclusion

In sum we considered how a small customer participating in a CP pricing program can near-optimally trade off lost revenue for CP cost savings. We formulated an approximate dynamic programming problem that incorporates successive observations of system loads likely to be a CP as inputs allowing a customer subject to ramping constraints to make informed curtailment decisions over the course of the billing period time horizon. This work improves on existing algorithms such as [4] or [8] by escaping an ad-hoc threshold curtailment regime where if some measure exceeds a threshold parameter than the customer should curtail. Further, this optimization framework provides footing for further theoretic analysis, detailed below.

A. Future Work

The goal of future work is to explore the implications of the optimization formulation in (3) for large players as observations of S_t become a function of x_t. Multiple large customers contributing to S_t resemble a Cournot competition, and a desirable outcome might be the existence of convergent strategies for large customers.

Additionally, for both large and small players, given a customer's time-coupled revenue function, how are they incentivized to participate in a CP program? Intuitively a customer with more demand flexibility—such as in the case of the data center in [4]—has potentially more to gain from participating in a CP pricing program than a comparably sized customer with little demand flexibility. If this is the case, how do factors such as CP billing horizon affect incentives, e.g. annually vs. monthly, impact this incentives like discounted time of use rates in exchange for participating?

References

[1] Moslem Uddin, Mohd Fakhizan Romlie, Mohd Faris Abdullah, Syahirah Abd Halim, Tan Chia Kwang, et al., "A review on peak load shaving strategies," *Renewable and Sustainable Energy Reviews*, vol. 82, pp. 3323–3332, 2018.

[2] ERCOT, "Electrical reliability council of texas, four coincident peak calculations," `http://www.ercot.com/mktinfo/data_agg/4cp`, 2018, Accessed: 2018-06.

[3] Fort Collins PUD, "City of fort collins utilies, 2018 rates for large commerical consumers," `https://www.fcgov.com/utilities/img/site_specific/uploads/Large_Commercial_2018_Rates_Brochure1.pdf`, 2018, Accessed: 2018-06.

[4] Zhenhua Liu, Adam Wierman, Yuan Chen, Benjamin Razon, and Niangjun Chen, "Data center demand response: Avoiding the coincident peak via workload shifting and local generation," *Performance Evaluation*, vol. 70, no. 10, pp. 770–791, 2013.

[5] Silicon Valley Power, "Stakeholder comments review tac structure straw proposal," `https://www.caiso.com/Documents/SVPComments-ReviewTransmissionAccessChargeStructure-StrawProposal.pdf`, 2018.

[6] Jay Zarnikau and Dan Thal, "The response of large industrial energy consumers to four coincident peak (4cp) transmission charges in the texas (ercot) market," *Utilities Policy*, vol. 26, pp. 1–6, 2013.

[7] ERCOT, "Electrical reliability council of texas, long term load forecast," `http://www.ercot.com/gridinfo/load/forecast/2017`, 2018, Accessed: 2018-06.

[8] Chase P Dowling, Daniel Kirschen, and Baosen Zhang, "Coincident peak prediction using a feed-forward neural network," in *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE, 2018, pp. 912–916.

[9] Jay Zarnikau, Greg Landreth, Ian Hallett, and Subal C Kumbhakar, "Industrial customer response to wholesale prices in the restructured texas electricity market," *Energy*, vol. 32, no. 9, pp. 1715–1723, 2007.

[10] Adam Wierman, Zhenhua Liu, Iris Liu, and Hamed Mohsenian-Rad, "Opportunities and challenges for data center demand response," in *International Green Computing Conference*. IEEE, 2014, pp. 1–10.

[11] Yuanyuan Shi, Bolun Xu, Baosen Zhang, and Di Wang, "Leveraging energy storage to optimize data center electricity cost in emerging power markets," in *Proceedings of the Seventh International Conference on Future Energy Systems*. ACM, 2016, p. 18.

[12] Cheng Wang, Bhuvan Urgaonkar, Qian Wang, George Kesidis, and Anand Sivasubramaniam, "Data center power cost optimization via workload modulation," in *Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*. IEEE Computer Society, 2013, pp. 260–263.

[13] Daniel S Kirschen and Goran Strbac, *Fundamentals of power system economics*, John Wiley & Sons, 2018.

[14] Lillian J Ratliff, Roy Dong, Henrik Ohlsson, and S Shankar Sastry, "Incentive design and utility learning via energy disaggregation," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 3158–3163, 2014.

[15] Eugene A Feinberg and Dora Genethliou, "Load forecasting," in *Applied mathematics for restructured electric power systems*, pp. 269–285. Springer, 2005.

[16] Rafal Weron, *Modeling and forecasting electricity loads and prices: A statistical approach*, vol. 403, John Wiley & Sons, 2007.

[17] Jennie Si, Andrew G Barto, Warren B Powell, and Don Wunsch, *Handbook of learning and approximate dynamic programming*, vol. 2, John Wiley & Sons, 2004.

[18] Dimitri P Bertsekas and John N Tsitsiklis, *Neuro-dynamic programming*, vol. 5, Athena Scientific Belmont, MA, 1996.

[19] Fei-Yue Wang, Huaguang Zhang, Derong Liu, et al., "Adaptive dynamic programming: An introduction," 2009.