Online Peak-Aware Energy Scheduling with Untrusted Advice

Russell Lee
UMass Amherst
rclee@cs.umass.edu

Jessica Maghakian
Stony Brook University
jessica.maghakian@stonybrook.edu

Mohammad Hajiesmaili
UMass Amherst
hajiesmaili@cs.umass.edu

Jian Li
Binghamton University-SUNY
lij@binghamton.edu

Ramesh Sitaraman
UMass Amherst and Akamai Tech
ramesh@cs.umass.edu

Zhenhua Liu
Stony Brook University
zhenhua.liu@stonybrook.edu

ABSTRACT

This paper studies the online energy scheduling problem in a hybrid model where the cost of energy is proportional to both the volume and peak usage, and where energy can be either locally generated or drawn from the grid. Inspired by recent advances in online algorithms with Machine Learned (ML) advice, we develop parameterized deterministic and randomized algorithms for this problem such that the level of reliance on the advice can be adjusted by a trust parameter. We then analyze the performance of the proposed algorithms using two performance metrics: *robustness* that measures the competitive ratio as a function of the trust parameter when the advice is inaccurate, and *consistency* for competitive ratio when the advice is accurate. Since the competitive ratio is analyzed in two different regimes, we further investigate the Pareto optimality of the proposed algorithms. Our results show that the proposed deterministic algorithm is Pareto-optimal, in the sense that no other online deterministic algorithms can dominate the robustness and consistency of our algorithm. Furthermore, we show that the proposed randomized algorithm dominates the Pareto-optimal deterministic algorithm. Our large-scale empirical evaluations using real traces of energy demand, energy prices, and renewable energy generations highlight that the proposed algorithms outperform worst-case optimized algorithms and fully data-driven algorithms.

CCS CONCEPTS

• **Theory of computation** → **Online algorithms**; **Linear programming**; • **Hardware** → *Energy generation and storage*; *Power estimation and optimization.*

KEYWORDS

Machine learned advice, online algorithms, renewable generation, Pareto-optimality

ACM Reference Format:
Russell Lee, Jessica Maghakian, Mohammad Hajiesmaili, Jian Li, Ramesh Sitaraman, and Zhenhua Liu. 2021. Online Peak-Aware Energy Scheduling with Untrusted Advice. In *The Twelfth ACM International Conference on Future Energy Systems (e-Energy '21), June 28–July 2, 2021, Virtual Event, Italy.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3447555.3464860

e-Energy '21, June 28–July 2, 2021, Virtual Event, Italy

ACM ISBN 978-1-4503-8333-2/21/06…$15.00
https://doi.org/10.1145/3447555.3464860

1 INTRODUCTION

The electricity bill is a significant operating cost of large energy customers such as data centers, businesses, and university campuses. For example, in data center operations, the largest expenditure is energy consumption, e.g. energy cost is more than 30% of the total operating costs of Google and Microsoft's data centers [25]. Consequently, managing the energy consumption and cost of large energy customers has become critically important. This has led to substantial research on incorporating local renewable sources [19], energy-aware server provisioning [18], geographical load balancing [17, 20], and on-site energy storage systems [31].

The electricity bill for large energy customers is usually based on a hybrid model that uses both the volume and peak of the energy consumption. Specifically, assuming that each billing cycle can be divided into T time slots, and the energy demand in slot t is $e(t)$, the electricity bill is the sum of the following two terms: (1) the *volume pricing*, which is the aggregate energy usage over the cycle, i.e., $\sum_t p(t)e(t)$, where $p(t)$ is the real-time unit price at t, and (2) the *peak pricing*, which is the peak demand drawn over the cycle, i.e., $\max_{t\in[T]} e(t)$, multiplied by p_m as the peak price. The contribution of peak pricing in the electricity bill is usually substantial. The peak price is often more than 100 times higher than the maximum spot price, e.g., 118× for PG&E or 227× for Duke Energy Kentucky. Hence, the contribution of peak charging in the energy bill for large energy costumers can be considerable, e.g., from 20% to 80% for Google data centers [30].

A promising approach to reduce the contribution of peak charges in the final electricity bill is to install on-site generation units that can "shave the peak" by covering that portion of the demand [33]. A notable example is Microsoft's plan to add 72 new generators at its Quincy, Washington data center campus [1]. The global market for on-site generators is growing and expected to reach a revenue of around $5 billion in 2023 [2]. With an on-site generator, one can schedule its generation such that part of the total energy demand is satisfied by the local generator, hence, the peak net demand from the grid is reduced over the billing cycle.

However, peak-aware energy generation scheduling of local generators is a challenging problem due to the uncertainty of the demand of energy customers, especially in data centers. For data centers the energy demand is highly unpredictable because user demand for internet services is variable. For instance, a data center serving videos to users can experience an unexpected flash crowd of users for a popular video release. Furthermore, sophisticated

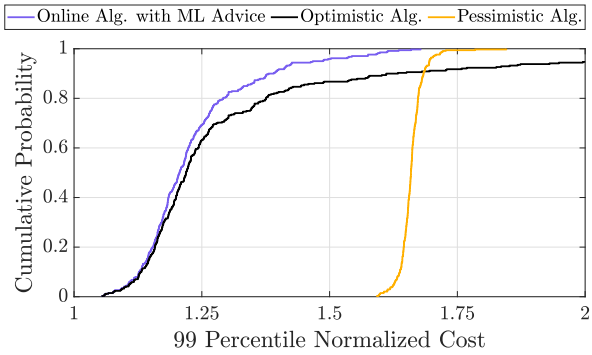


Figure 1: A motivating example for the peak-aware energy scheduling problem.

optimization algorithms are being used in Google data centers to improve the energy efficiency of data center's internal operations [12], which can further increase the variability of energy demand. In geographical load balancing schemes [17, 20], a global load balancer could move user demand into or out of the data center, resulting in unexpected changes in the energy patterns. Lastly, the integration of renewables into data centers provides even more uncertainty, since the production level of renewables is uncertain and intermittent [13].

The peak-aware energy generation scheduling problem (henceforth PAES) has been tackled using the competitive online framework [33]. More specifically, two deterministic and randomized algorithms have been proposed that can achieve the best *competitive ratio* as the well-established performance metric for online algorithms [6]. Competitive ratio is defined as the ratio between the cost of an online algorithm and that of the offline optimal for the worst-case over all feasible instances to the problem. The competitive online framework, however, aims to be provably efficient against worst-case input instances. Toward this end, it assumes that no stochastic, exact, or noisy measurement of future inputs is available and tries to make the best decisions without future knowledge. This assumption makes online algorithms overly pessimistic in practice, since worst-case scenarios rarely happen in reality. On top of that, it is reasonable to have at least a noisy prediction of future data in most online problems.

1.1 Motivation

As a motivating example, we consider the performance of online algorithms designed with three different paradigms. Figure 1 shows the normalized cost achieved by three algorithms for PAES. The three online algorithms shown are: (a) the algorithm that achieves the best competitive ratio for "pessimistic" worst-case inputs for the energy cost minimization problem [33]; (b) the algorithm that makes decisions assuming an "optimistic" world of perfect predictions; and (c) a proposed online algorithm in this paper that aims to perform well in both worlds. The figure shows the cumulative probability distribution of the 99 percentile normalized cost of 100+ trials for traffic and energy inputs from 200+ Akamai data center locations in the United States (details on experimental data and setting in §6). Although the state-of-the-art "pessimistic" online algorithm is guaranteed to achieve a bounded normalized cost in the worst-case, Figure 1 demonstrates that its average performance is not promising. The average normalized cost is far from optimal, with even the best performing trials achieving normalized cost no less than 1.6. Conversely, an "optimistic" data-driven online algorithm has better average performance, with about 60% of trials having a normalized cost of 1.25 or lower. As a trade-off, it has a heavy tail of worst-case instances where the normalized cost is worse than 2. However, online algorithms with ML advice (the algorithmic approach of this work) achieve the best of both worlds: good average performance as well as the best guarantee for the worst-case tail performance through prudent usage of predictions spanning between pure optimism and pure pessimism.

1.2 Algorithmic Approach

The goal of this paper is to design competitive algorithms with advice for the PAES problem. Our approach is inspired by the recent effort on integrating machine learned (ML) advice to improve the practical performance of online algorithms [14, 16, 22, 24]. The key motivation is two-fold: (1) to keep the core competency of online algorithms, i.e., performance guarantee against the worst-case; and (2) to achieve a provably improved performance if the accuracy of ML-predictor is satisfactory. The two motivations could be analyzed for *learning-assisted online algorithms* [22, 24] by introducing the notions of (1) *robustness* that characterizes the first motivation; and (2) *consistency* that characterizes the second one.

Specifically, suppose that $\mathcal{A}$ is a learning-assisted online algorithm that leverages an ML-predictor in decision making. The algorithm $\mathcal{A}$ is (α, γ)-competitive where α and γ represent the robustness and consistency of $\mathcal{A}$, respectively. That is, the competitive ratio of $\mathcal{A}$ is always less than α regardless of the error in ML-predictor. Also, $\mathcal{A}$ is γ-*consistent* if with perfect predictions it achieves the competitive ratio of γ. Robustness measures how well the algorithm does in the worst-case of poor predictions, while consistency measures how well the algorithm does under perfect predictions. In this framework, the performance of an algorithm is evaluated using two criteria, i.e., robustness and consistency. Hence, investigating the optimality of an algorithm naturally leads to the consideration of Pareto optimality. Therefore, the eventual goal in this setting is to design an algorithm $\mathcal{A}$ that is Pareto-optimal, meaning that there is no other algorithm that can achieve a better consistency (resp., robustness) than $\mathcal{A}$ without sacrificing the robustness (resp., consistency).

With this analytical framework, one is able to achieve "the best of both worlds" paradigm from the perspective of learning-assisted competitive algorithms. While it might slightly degrade the robustness against worst-case, or ideally maintain the worst-case guarantee, it resolves the fundamental drawback of competitive analysis of pessimistic decision making by incorporating ML predictions. More importantly, unlike classic prediction-based competitive designs [7, 9, 10, 15], the framework used in this paper leverages a *trust parameter* that determines how much the algorithms trust the predictors, enabling the full spectrum coverage from pure worst-case to fully prediction-based decision making.

Table 1: A summary of theoretical results. Here $\lambda \in (0, 1]$ is the trust parameter, $\beta \in (0, 1)$ is a problem-specific parameter, and Φ_1, Φ_2 are expressions for the randomized algorithm defined in Algorithm 3.

Algorithm	Theoretical results	Property
OnMLEng (Deterministic)	**Robustness:** $1 + (1-\beta)/\lambda$ **Consistency:** $1 + \lambda(1-\beta)$	OnMLEng is Pareto optimal
rOnMLEng (Randomized)	**Robustness:** $\max\{1+\Phi_1(1-\beta), 1+\Phi_2(1-\beta) [(e^{1/\lambda} - 1 - 1/\lambda)(1/\lambda - 1) + 1/\lambda^2]\}$ **Consistency:** $\max\{1+\Phi_1\lambda^2(1-\beta), 1+\Phi_2(1-\beta)\}$	rOnMLEng dominates the Pareto optimal deterministic algorithm

1.3 Summary of Contributions

Inspired by the above direction of learning-assisted algorithm design, we develop deterministic and randomized algorithms for PAES that take into account advice from an ML model in decision making. This paper makes the following contributions, with a summary of theoretical results outlined in Table 1.

First, we propose OnMLEng, a deterministic algorithm parameterized by a trust parameter $\lambda \in (0, 1]$, that achieves a competitive ratio of $1 + (1-\beta)/\lambda$, where $\beta \in (0, 1)$ is a problem-specific parameter that determines the ratio between the unit price of the grid and local generator. We show that OnMLEng is $(1+(1-\beta)/\lambda)$-robust and $(1 + \lambda(1-\beta))$-consistent. The trust in ML prediction is interpreted as follows. Greater trust in ML prediction is achieved by setting λ close to 0, which means that OnMLEng is 1-consistent, i.e., it achieves the optimal performance with perfect advice. On the other hand, less trust in ML advice is achieved by setting λ close to 1, and the robustness result guarantees the optimal online competitive ratio of $2 - \beta$ as in [33]. More importantly, we show that OnMLEng is Pareto-optimal, showing that our deterministic algorithm achieves the best possible robustness and consistency bounds.

Second, we propose rOnMLEng, a randomized algorithm with a trust parameter λ that has both robustness and consistency guarantees. With $\lambda = 1$, rOnMLEng recovers the competitive ratio of the best randomized algorithm with the competitive ratio of $e/(e-1+\beta) \leq 1.58$. With $\lambda \to 0$, rOnMLEng is 1-consistent, i.e., it behaves optimally. The design and analysis of rOnMLEng is a significant theoretical contribution of this paper. Specifically, it is worth noting that the probability distribution functions of rOnMLEng are carefully designed to achieve solid robustness and consistency guarantees. These distribution functions are customized based on Yao's principle [32] to provide robustness and consistency results in a more systematic manner compared to the randomized algorithm design for online problems in [16, 22, 24]. Finally, we show that rOnMLEng dominates the Pareto-optimal deterministic algorithm OnMLEng.

Last, we empirically evaluate the performance of the algorithms using real-world data traces. We use energy demand traces from Akamai data centers [23] as an example of large-scale industrial load, as well as energy price values from New York energy market (NYISO). The results show the improved performance of our proposed online algorithms with ML advice as compared to the purely online algorithm. We also investigate the impact of several parameters and provide insights that reveal the practical benefits of learning-assisted algorithms.

2 PROBLEM STATEMENT

We consider the scenario where the energy demand can be covered by either local generators or the external grid. The peak-aware energy scheduling problem (PAES) aims to prudently choose the source of energy, so that the energy demand can be met at each time step while the total cost is minimized.

We focus on one billing cycle $\mathcal{T} = \{1, \cdots, T\}$ with T discrete time slots of uniform length. The billing cycle is usually one month and the length of each slot is 5 minutes. Let the energy demand in slot t be $e(t)$ and $\boldsymbol{e} = [e(t)]_{t\in[T]}$. We consider an online scenario in which the values of demand are unknown for future slots. The demand can be covered by two sources, the *local generator* and *external grid*. The local generator can satisfy at most $C \geq 1$ KWs of demand in each slot, with cost p_g. In reality, some traditional generators [21] have maximum ramp-up and ramp-down constraints that limit the change of output between two adjacent slots. In this paper, we focus on "fast-responding" generators that can ramp up and down without any limit. In experiments (§6.3.3), we investigate the impact of ramp constraints.

We consider a typical energy cost model for industrial energy customers that follows a hybrid charging model that has both total usage (a.k.a. energy charge) and peak usage (a.k.a. demand charge) components. The energy cost is the sum of the following two terms: (1) the *usage-based pricing*, which is the total energy usage over the cycle, and (2) the *peak pricing*, which is the peak demand drawn over the cycle. Following the dynamics of the energy market, the grid provides electricity with a spot price $p(t)$ at time t, where we assume $p(t) \geq p^{\min} > 0$. In reality, the unit cost of local generators p_g is usually higher than that of external grid, i.e., $p_g \geq p(t)$. Otherwise, it is always optimal to use local generators as much as possible for both online and offline algorithms. However, the expensive local generator can shave the peak demand (peak charge) of the external grid. In addition, p_m is the peak charge price that is known and fixed over the billing cycle. Note that p_m is usually more than 100 times larger than $p(t)$. For ease of exposition, denote $\beta \triangleq p^{\min}/p_g < 1$ as the ratio between the minimum grid price and the unit cost of local generation. We characterize the performance of our algorithms as a function of β.

Let $v(t)$ and $u(t)$ be the optimization variables that determine the amount of electricity procured from the external grid and local generator, respectively. For the grid, its cost consists of volume charge and peak charge. The volume charge is the sum of volume cost over the time horizon, i.e., $\sum_t p(t)v(t)$. The peak charge is based on the maximum single-slot power and peak price p_m, i.e., $p_m \max_t v(t)$ [30, 33]. The cost of using local generators, is $\sum_t p_g u(t)$. Therefore, with $\boldsymbol{u} = [u(t)]_{t\in\mathcal{T}}$ and $\boldsymbol{v} = [v(t)]_{t\in\mathcal{T}}$, the PAES problem is defined as

$$\begin{aligned} \text{PAES}: \quad \min_{\boldsymbol{u},\boldsymbol{v}} \quad & \sum_{t\in\mathcal{T}} p(t)v(t) + p_m \max_t v(t) + \sum_{t\in\mathcal{T}} p_g u(t) \\ \text{s.t.}, \quad & u(t) + v(t) \geq e(t), \quad t \in \mathcal{T}, \\ & u(t) \leq C, \quad t \in \mathcal{T}, \end{aligned}$$

$$v(t) \geq 0, u(t) \geq 0, \quad t \in \mathcal{T},$$

where the first constraint ensures that the demand is satisfied, and the second constraint is due to the generator capacity limitation. We note that in our algorithm design we focus on a basic version of PAES, where the demand $e(t)$ only takes binary values 0 or 1. Our algorithms and competitive analysis, however, could be extended to the general case as discussed in Section 4.6.

PAES with $e(t)$ and $p(t)$ values known in advance is a linear program. Hence, it can be solved using any linear programming algorithm. However, in practice $e(t)$ and $p(t)$ are unknown in advance and hard to predict, hence an online approach is required. We use a recently proposed algorithmic framework [22, 24] for devising online algorithms with advice, and provide a brief overview of the framework in the following section.

3 ALGORITHMIC FRAMEWORK

In this section, we give an overview of the recently proposed framework for designing competitive algorithms with ML advice [22, 24]. In this framework the goal is to utilize ML advice to improve the performance of online algorithms, both in theory and practice. Toward this, it is assumed that there is advice from an untrusted ML model that might be subject to error or even vulnerable to malicious activities. The goal is to develop online algorithms that are able to determine the level of trust in the ML advice.

Trust. The trust parameter determines how much the algorithm trusts the ML advice. More formally, let $\lambda \in (0, 1]$ be a trust parameter that indicates the level of trust that we place on the advice. In our algorithms, setting $\lambda \to 0$ represents full trust in ML advice, and $\lambda \to 1$ indicates no trust at all, i.e., making worst-case decisions similar to the classic competitive framework. Any value in between indicates partial trust in ML advice.

Robustness and Consistency. The performance of an algorithm in this framework is captured using two metrics that reflect two extreme cases when the advice is inaccurate and when the advice is fully accurate. Specifically, suppose that $\mathcal{A}_\lambda$ is an online algorithm that leverages ML advice in decision making with the trust parameter λ. Let ϵ be the error of the ML advice, which is the absolute difference between the advice and actual outcome. Denote ALG(ϵ, λ) as the cost of $\mathcal{A}_\lambda$ given λ as the trust parameter and ϵ as the error of the ML advice, and OPT as the offline optimum, respectively.

Definition 1. *(Robustness) $\mathcal{A}_\lambda$ is α-robust if ALG$(\epsilon, \lambda) \leq \alpha \cdot$ OPT for all ϵ and feasible instances to the problem.*

Definition 2. *(Consistency) $\mathcal{A}_\lambda$ is γ-consistent if ALG$(0, \lambda) \leq \gamma \cdot$ OPT when the ML advice is accurate ($\epsilon = 0$) and for all feasible instances to the problem.*

Note that α and γ could be functions of the problem parameters as well as λ and ϵ. Intuitively, robustness measures how well the algorithm does in the worst-case of poor advice, and consistency measures how well the algorithm does with perfect advice.

Pareto Optimality. In the traditional framework with competitive ratio as the performance metric, the notion of *optimality* refers to an online algorithm that achieves *the best possible competitive ratio*. In the new framework with ML advice, the performance of algorithms is measured by two criteria: robustness and consistency. The superiority of an algorithm in a bi-criteria setting against an alternative can be measured using the notion of *dominance* and *Pareto optimality*.

Definition 3. *(Dominance) For comparing two online algorithms A and B, we say that A dominates B if it is better in both criteria, i.e., $\alpha_A \leq \alpha_B$ and $\gamma_A \leq \gamma_B$.*

With the trust parameter, we develop a class of online algorithms in which each instance of the algorithm refers to a specific value of trust parameter. Hence, for analyzing the dominance of the algorithms, our goal is to investigate the Pareto frontier properties of a class of algorithms.

Definition 4. *(Pareto Optimality) Let $\mathcal{A} = \{A_\lambda, \lambda \in (0, 1]\}$ be the class of online algorithms with trust parameter λ. $\mathcal{A}$ is Pareto-optimal if for any other online algorithm B, there exists $A_\lambda \in \mathcal{A}$ such that A_λ dominates B.*

In Section 4, we develop deterministic and randomized algorithms using the above framework and analyze their robustness, consistency, and Pareto-optimality in Section 5. Our proposed algorithms are built on top of existing fully online algorithms that do not use ML advice for decision making. We briefly introduce these algorithms in the following subsection.

Existing Online Algorithms without ML advice. The idea of prior online algorithms (`OnEng`) [33] lies in constructing a break-even point that balances between the cost of using generators and the peak charge of using the grid. Specifically, break-even point σ is

$$\sigma = \frac{1}{p_m}\Big[\sum_{t\in\mathcal{T}}(p_g - p(t))e(t)\Big]. \tag{1}$$

The parameter σ plays a critical role in algorithm design. For an optimal offline algorithm, we have $v^*(t) = e(t)$, $\forall t \in \mathcal{T}$, when $\sigma > 1$; and $v^*(t) = 0$, $\forall t \in \mathcal{T}$, otherwise. The optimal output of the local generator is then $u^*(t) = e(t) - v^*(t)$.

The value of σ can be calculated easily in an offline manner. However, with unknown price and demand values, this value cannot be fully computed online. The high-level idea of `OnEng` is make a decision based on a partially-calculated value of σ over the current and past slots. Specifically, `OnEng` keeps using the local generator initially and switches to the grid at the first time τ such that $\sum_{t=1}^{\tau}(p_g - p(t))e(t) \geq p_m$. The competitive ratio of `OnEng` is $2 - \beta$. The proof ideas are similar to the ski-rental problem and they show that the break-even point is the best balance between being aggressive (paying the one-time premium peak cost) and being conservative (using the local generator). Finally, the competitive ratio has been improved to $e/(e - 1 + \beta) \leq 1.58$, by developing a randomized algorithm (`rOnEng`), in which the algorithm starts purchasing grid electricity when $\sum_\tau(p_g - p(\tau)) \geq s \cdot p_m$, where s is chosen randomly according to the following distribution

$$f^*(s) = \begin{cases} \frac{e^s}{e-1+\beta}, & \text{when } s \in [0, 1]; \\ \frac{\beta}{e-1+\beta}\delta(0), & \text{when } s = \infty; \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

4 ALGORITHM DESIGN

In this section, we first introduce how the ML advice could be constructed for the PAES problem, then present a Pareto-optimal deterministic algorithm, and finally a randomized algorithm that dominates our proposed Pareto-optimal deterministic algorithm. We also highlight our technical results. The detailed derivation of the theoretical results are given in Section 5.

4.1 ML advice for the PAES problem

First, we introduce the ML advice. Assume that there is a learning model that predicts the future values of external grid prices, $\hat{p}(t)$, and energy demand, $\hat{e}(t)$. The key idea in our online algorithm design lies in constructing a break-even point using these two values so as to balance between the cost of using generators and the peak charge of using the grid. Given these two values, let $\hat{\sigma}$ be the predicted break-even point using the learning model as

$$\hat{\sigma} \triangleq \frac{1}{p_m}\Big[\sum_{t\in\mathcal{T}}(p_g - \hat{p}(t))\hat{e}(t)\Big]. \tag{3}$$

Note that it is even possible that the ML model directly predicts the value of $\hat{\sigma}$ based on historical break-even points. Hence, predicting individual values of $\hat{p}(t)$ and $\hat{e}(t)$ for the cycle is not needed.

4.2 A Simple Consistent and Non-Robust Algorithm

We first show that there exists an algorithm `Eng-dd` for PAES that naively uses the predicted break-even point and is 1-consistent, i.e., its competitive ratio is 1 when the advice is accurate. However, it is straightforward to show that this algorithm is not robust since the competitive ratio can be arbitrarily large in the case of incorrect predictions. We empirically compare the result of our robust and consistent algorithms with this simple algorithm in Section 6.

Algorithm 1 `Eng-dd`

if $\hat{\sigma} > 1$ **then**
  use the local generator entirely
else
  use the grid entirely
end if

4.3 `OnMLEng`: A Deterministic Robust and Consistent Algorithm

We propose an online algorithm `OnMLEng` with ML advice that uses the trust parameter $\lambda \in (0, 1]$ to determine the level of trust in advice as introduced in Section 3. `OnMLEng` makes decisions based on the values of $\hat{\sigma}$ and λ as summarized in Algorithm 2. Note that in `OnMLEng`, $\lambda \to 0$ (full trust) is equivalent to running `Eng-dd`.

Theorem 1. *The* `OnMLEng` *algorithm achieves the competitive ratio of* $1 + (1-\beta)/\lambda$, *where* $\lambda \in (0, 1]$.

Corollary 1. `OnMLEng` *is* $(1 + (1-\beta)/\lambda)$*-robust.*

Corollary 2. `OnMLEng` *is* $(1 + \lambda(1-\beta))$*-consistent.*

Algorithm 2 `OnMLEng`

if $\hat{\sigma} > 1$ **then**
  $s \leftarrow \lambda$
else
  $s \leftarrow 1/\lambda$
end if
Use local generator first and commit to switching to the grid starting at the first time τ where

$$\sum_{t=1}^{\tau}(p_g - p(t))e(t) \geq s \cdot p_m.$$

Algorithm 3 `rOnMLEng`

Denote $\Phi_1 = \frac{1}{e^{\lambda}-1+\lambda^2\beta}$ and $\Phi_2 = \frac{1}{e^{\frac{1}{\lambda}}-1+\frac{1}{\lambda^2}\beta}$

if $\hat{\sigma} > 1$ **then**

$$f_1^*(s) = \begin{cases} \Phi_1 e^s, & s \in [0, \lambda]; \\ \Phi_1\lambda^2\beta\delta(0), & s = \infty; \\ 0, & \text{otherwise.} \end{cases}$$

else

$$f_2^*(s) = \begin{cases} \Phi_2 e^s, & s \in [0, \frac{1}{\lambda}]; \\ \Phi_2\frac{1}{\lambda^2}\beta\delta(0), & s = \infty; \\ 0, & \text{otherwise.} \end{cases}$$

end if
Pick a value s randomly according to probability distribution $f_1^*(s)$ or $f_2^*(s)$, and switch to grid electricity starting at the first time τ where

$$\sum_{t=1}^{\tau}(p_g - p(t))e(t) \geq s \cdot p_m.$$

Intuitively, $s \in (0, \infty)$ is a function of $\hat{\sigma}$ and λ that determines when `OnMLEng` switches to the grid. Setting $\lambda = 1$ will fix $s = 1$, which recovers the robustness competitive ratio of $2 - \beta$ from the optimal online algorithm `OnEng` [33]. This implies that with bad advice it suffices to completely distrust the advice to be robust against worst-case performance. On the other hand, setting $\lambda \to 0$ will result in $s \to 0$ or $s \to \infty$, which means immediately switching to the grid or entirely staying with the local generator respectively. This results in a consistency of 1. Tuning the value of λ effectively adjusts the level of trust in advice by determining s.

Next, in Theorem 2, we show that `OnMLEng` represents a family of Pareto-optimal algorithms specified by the trust parameter λ, based on Definition 4.

Theorem 2. `OnMLEng` *defines the Pareto frontier of robustness and consistency for the* PAES *problem, and is Pareto-optimal for all deterministic algorithms that solve* PAES.

The above result shows that `OnMLEng` defines the Pareto frontier. In other words, there is no other family of deterministic algorithms that can achieve a better consistency (resp., robustness) than `OnMLEng` without sacrificing the robustness (resp., consistency).

Furthermore, we show that for any deterministic algorithm $\mathcal{A}$ that solves PAES, it can be expressed by a deterministic algorithm with a switching parameter, i.e., `OnMLEng` is Pareto-optimal for any deterministic algorithm $\mathcal{A}$.

4.4 rOnMLEng: A Randomized Robust and Consistent Algorithm

In randomized algorithms, the decision making is based on random variable draws from a proper probability distribution function. We develop a randomized algorithm, rOnMLEng, as summarized in Algorithm 3. In rOnMLEng we modify the probability distribution function of rOnEng [33] based on λ and $\hat{\sigma}$ as in Eq. (3).

Theorem 3. rOnMLEng *achieves a competitive ratio of*

$\max\{1+\Phi_1(1-\beta), 1+\Phi_2(1-\beta)[(e^{1/\lambda}-1-1/\lambda)(1/\lambda-1)+1/\lambda^2]\}$,

where $\lambda \in (0,1]$, $\Phi_1 = \frac{1}{e^{\lambda}-1+\lambda^2\beta}$ *and* $\Phi_2 = \frac{1}{e^{1/\lambda}-1+1/\lambda^2\cdot\beta}$.

Corollary 3. rOnMLEng *is* $(\max\{1+\Phi_1(1-\beta), 1+\Phi_2(1-\beta)[(e^{1/\lambda}-1-1/\lambda)(1/\lambda-1)+1/\lambda^2]\})$*-robust.*

Corollary 4. rOnMLEng *is* $(\max\{1+\Phi_1\lambda^2(1-\beta), 1+\Phi_2(1-\beta)\})$*-consistent.*

Corollary 5. *The consistency and robustness bounds of* rOnMLEng *are strictly better than those of* OnMLEng.

The probability distributions $f_1^*(s), f_2^*(s)$ are designed to satisfy two critical conditions. First, setting $\lambda = 1$ retains the original distribution function of the optimal randomized online algorithm rOnEng, and therefore would recover its competitive ratio of $e/(e-1+\beta)$. Second, setting $\lambda \to 0$ guarantees picking $s = 0$ or $s = \infty$ depending on the advice driven break-even point $\hat{\sigma}$. This will result in a competitive ratio of 1 for consistency, meaning matching optimal performance once the ML advice is accurate. Note that this follows the same selection of s in OnMLEng when $\lambda \to 0$.

Corollary 5 shows that the proposed randomized algorithm rOnMLEng dominates OnMLEng, the Pareto optimal deterministic algorithm. Lastly, in Appendix B we show that the randomized algorithm that naïvely modifies the distribution function of OnEng (2) based on the guidelines in deterministic algorithm fails to achieve satisfactory robustness and consistency at the same time.

4.5 OnMLEng-dyn and rOnMLEng-dyn: Dynamic Break-even Point Algorithms

OnMLEng and rOnMLEng utilize a static predicted break-even point $\hat{\sigma}$ that persists over the entire billing cycle, but our results can also be extended to utilizing a set of dynamic break-even points $\hat{\boldsymbol{\sigma}} = \{\hat{\sigma}_1, \hat{\sigma}_2, ..., \hat{\sigma}_T\}$. Having a dynamic break-even point captures a broad range of algorithms and allows a rich design space within OnMLEng and rOnMLEng. For example, predictions can be adjusted and possibly improved according to observed values over time. Algorithms that use a sliding window of predictions also fit within this framework, since the break-even point is dynamically calculated according to the available predictions.

Define OnMLEng-dyn as the version of OnMLEng where the set of predictions at each time step may change over time. In other words, let $\hat{\mathbf{e}}_\tau = [\hat{e}_\tau(t)]_{t\in[T]}, \hat{\mathbf{p}}_\tau = [\hat{p}_\tau(t)]_{t\in[T]}$, be the set of predictions for demand and grid price at time $\tau \in [1,T]$. Then the advice is dynamically provided as

$$\hat{\sigma}_\tau = \frac{1}{p_m}\Big[\sum_{t\in\mathcal{T}}(p_g - \hat{p}_\tau(t))\hat{e}_\tau(t)\Big], \tag{4}$$

and the decision to switch from the local generator to the grid is made according to $\hat{\sigma}_\tau$ and λ. Similarly, define rOnMLEng-dyn as the version of rOnMLEng using a set of break-even points $\boldsymbol{\sigma}$, with all else remaining the same. The detailed forms of OnMLEng-dyn and rOnMLEng-dyn can be found in Appendix C.

Theorem 4. *The robustness and consistency of* OnMLEng-dyn *and* rOnMLEng-dyn *are equivalent to those of* OnMLEng *and* rOnMLEng, *respectively.*

While the theoretical bounds of dynamic break-even algorithms are the same for robustness, consistency, and Pareto optimality, these algorithms are of practical importance because they can capture scenarios such as a sliding window of available predictions or improved prediction quality over time. We empirically evaluate the performance of one such algorithm in Section 6.

4.6 Extending Algorithms for Energy Problem to the General Case

The proposed algorithms for PAES in the paper are analyzed for a basic version in which the demand takes binary values of 0 or 1. Also, the corresponding competitive analyses are dedicated to the basic setting. However, the results can be straightforwardly extended to the general problem of non-negative integer demand. This is done by dividing the integer demand $e(t)$ into multiple sub-problems with binary demand. At a given layer i, the layered demand at time t is 1 if $e(t) \le i$ and 0 otherwise. Then the result in [33, Theorem 3] can be applied. By using the layered sub-problems strategy, the competitive ratio of an algorithm which solves the sub-problem with binary demand is an upper bound to the competitive ratio of an algorithm which solves the general integer demand problem. Similarly, the robustness and consistency of the binary demand setting provide an upper bound to the robustness and consistency of the general setting. Further proof and discussion on extending to the general case is in Appendix E.

5 PROOFS OF MAIN RESULTS

In this section, we provide the main proofs for the algorithms. The proofs for the competitive ratio of the randomized algorithm and dynamic algorithms are given in Appendix A and C.

5.1 Proof of Theorem 1

We analyze the competitiveness of OnMLEng. Given the structure of Algorithm 2, we can parameterize any online algorithm by parameter s. Let $\mathcal{A}_s$ be an online algorithm with a specific parameter s, e.g., OnMLEng is in this category with the value of s as in Algorithm 2. Let $h(\mathcal{A}_s, \sigma)$ be the ratio between the cost of algorithm $\mathcal{A}_s$ and that of an optimal offline algorithm given σ. The following proposition characterizes the closed-form value of $h(\mathcal{A}_s, \sigma)$, and facilitates the analysis of the proposed algorithm.

Proposition 1. [33] *For any online algorithm* $\mathcal{A}_s$*, we have*

$$\textit{when } \sigma \le 1, \quad h(\mathcal{A}_s,\sigma) = \begin{cases} 1, & \textit{if } s > \sigma; \\ 1+\frac{1-\sigma+s}{\sigma}(1-\beta), & \textit{otherwise.} \end{cases}$$

$$\textit{when } \sigma > 1, \quad h(\mathcal{A}_s,\sigma) = \begin{cases} 1+\frac{(\sigma-1)(1-\beta)}{(\sigma-1)\beta+1}, & \textit{if } s > \sigma; \\ 1+\frac{s(1-\beta)}{(\sigma-1)\beta+1}, & \textit{otherwise.} \end{cases}$$

We proceed to prove the robustness and consistency results. We first consider the robustness. The worst-case cost ratio for a general deterministic algorithm $\mathcal{A}_s$ with parameter s is when $\sigma = s$, where the online algorithm pays for the peak charge premium but has no net demand to serve anymore. From Proposition 1, this worst case cost ratio $\max_\sigma h(\mathcal{A}_s, \sigma)$ is

$$\max_{\sigma} h(\mathcal{A}_s, \sigma) = \begin{cases} 1 + \frac{1}{s}(1-\beta), & \text{if } s \le 1; \\ 1 + \frac{s(1-\beta)}{(s-1)\beta+1}, & \text{otherwise.} \end{cases} \tag{5}$$

We compute the competitive ratio of OnMLEng under two cases:

(i) $\hat{\sigma} > 1$: According to OnMLEng, $s = \lambda < 1$. From (5), we have $\mathrm{CR}(\mathcal{A}_\lambda) = 1 + (1-\beta)/\lambda$.

(ii) $\hat{\sigma} \le 1$: According to OnMLEng, $s = 1/\lambda > 1$. From (5), we have $\mathrm{CR}(\mathcal{A}_{1/\lambda}) = 1 + \frac{(1-\beta)/\lambda}{(1/\lambda-1)\beta+1}$.

This means that OnMLEng is $(1 + (1-\beta)/\lambda)$-robust. Note that setting $\lambda = 1$ recovers the competitive ratio of the optimal online algorithm.

Next, we consider the consistency. For consistency guarantees, we compute the competitive ratio assuming the predictions are correct. There are two cases to consider here:

(i) $\hat{\sigma} = \sigma > 1$, i.e., $s = \lambda$. From Proposition 1, when $\sigma > 1 \ge s = \lambda$, we have $\mathrm{CR}(\mathcal{A}_{1/\lambda}) = 1 + \frac{\lambda(1-\beta)}{(\sigma-1)\beta+1} \le 1 + \lambda(1-\beta)$.

(ii) $\hat{\sigma} = \sigma \le 1$, i.e., $s = 1/\lambda$. From Proposition 1, when $\sigma \le 1 \le s = 1/\lambda$, the worst case occurs when $s = 1/\lambda = \sigma$. Then $\mathrm{CR}(\mathcal{A}_{1/\lambda}) = 1 + \lambda(1-\beta)$.

This means that OnMLEng is $(1 + \lambda(1-\beta))$-consistent. Note that setting $\lambda \to 0$ results in a competitive ratio of 1, which means optimal performance with accurate predictions.

5.2 Proof of Theorem 2

First, we establish that OnMLEng is Pareto optimal for all deterministic algorithms with a switching parameter. From Theorem 1, OnMLEng is $(1 + \lambda(1-\beta))$-consistent and $(1 + \frac{1}{\lambda}(1-\beta))$-robust. Denote the consistency and robustness bounds as $\gamma_A = 1 + \lambda(1-\beta)$ and $\alpha_A = 1 + \frac{1}{\lambda}(1-\beta)$.

Consider an arbitrary algorithm A' that takes prediction-based advice with a consistency bound $\gamma_{A'}$ and robustness bound $\alpha_{A'}$. A' switches at either $i \cdot p_m$ or $j \cdot p_m$ based on the advice. Without loss of generality, we assume $i \le j$.

Lemma 1. *A' must be at least $\gamma_{A'} \ge \frac{1}{j}(1-\beta)$−consistent and $\alpha_{A'} \ge \frac{1}{j}(1-\beta)$-robust.*

Proof. We consider a couple of cases for the true break-even point σ utilizing Proposition 1: First, when $\sigma \le 1$, A' will either select $s = i$ or $s = j$ for competitive ratios of $1 + \frac{1}{i}(1-\beta)$ or $1 + \frac{1}{j}(1-\beta)$, respectively. We now consider the corresponding consistency and robustness:

(i) For consistency, A' has perfect predictions and knows exactly that $\sigma \le 1$. As a result, A' will rationally pick $s = j$, since $i \le j$ implies that $\frac{1}{j}(1-\beta) \le \frac{1}{i}(1-\beta)$. Then $\gamma_{A'} \ge \frac{1}{j}(1-\beta)$.

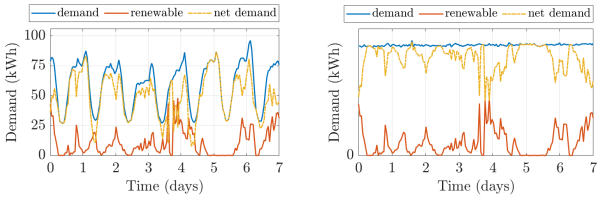
(ii) For robustness, A' does not have perfect predictions and cannot have full certainty that $\sigma \le 1$. Then A' could rationally pick $s = i$ or $s = j$, and in the worst case $s = i$ will be chosen. Then $\alpha_{A'} \ge \frac{1}{i}(1-\beta)$. □

Assume that A' has a lower consistency bound than OnMLEng, i.e. $\gamma_{A'} \le 1 + \lambda(1-\beta)$. It follows that $1 + i(1-\beta) \le 1 + \lambda(1-\beta)$, and subsequently $i \le \lambda$. Applying this to the robustness bound for $\alpha_{A'}$ yields $\alpha_{A'} \ge 1 + \frac{1}{i}(1-\beta) \ge 1 + \frac{1}{\lambda}(1-\beta) = \alpha_A$.

This concludes the proof that OnMLEng is Pareto optimal for all deterministic algorithms with a switching parameter, since $\gamma_{A'} \le \gamma_A$ guarantees that $\alpha_{A'} \ge \alpha_A$ for any algorithm A' with switching parameters i, j.

Lemma 2. *Any deterministic algorithm for PAES can be expressed by a deterministic algorithm with a switching parameter.*

The main idea for proving this lemma is that time slots assigned to the local generator and the grid can be reordered such that the assignment can be determined by a single parameter. The full details of the proof are given in Appendix D. Combining Lemma 1 and Lemma 2 concludes the proof of Theorem 2.



(a) New York City (b) Rochester

Figure 2: Time-varying energy demand with incorporation of renewables for two Akamai data centers in NY. Although there is a roughly diurnal pattern for energy demand in New York City, Rochester is comparably more unpredictable.

6 EXPERIMENTS

We use real-world traces to experimentally evaluate the performance of proposed learning-assisted algorithms as compared to the pure online algorithms and the offline optimum. Our proposed algorithms characterize a class of algorithms that are determined by the choice of trust parameter. Our experiments consider such algorithms in both the worst-case performance and practical average-case performance scenarios. The results answer these questions:

(1) *How does the* OnMLEng *algorithm compare to the pure online algorithm?* Our results show that OnMLEng consistently achieves better average performance than the pure online algorithm, sometimes even achieving near-optimality.
(2) *What is the effect of varying prediction quality via renewable penetration?* Lower-quality predictions noticeably degrade the worst-case performance of OnMLEng instances that are too optimistic about advice, while the performance of OnMLEng instances with more cautious trust parameter selection is robust to poorer prediction quality.
(3) *How do problem parameters such as peak price and grid capacity constraints affect the performance?* The normalized cost of the best performing OnMLEng algorithm remains extremely close to optimal under four different varying parameters.

Table 2: Summary of algorithms that are evaluated

The online algorithms with ML advice that we evaluate	
OnMLEng-opt	OnMLEng with the optimal trust parameter (offline)
OnMLEng-hist	OnMLEng with the best historical trust parameter (online)
OnMLEng-hist-dyn	OnMLEng-hist with time-varying predictions that are adjusted to align with observed values
Eng-dd	A simple data-driven algorithm for PAES that fully trusts the break-even point of the previous instance
Other algorithms for comparison	
ENG-OPT	Optimal offline cost for PAES
OnEng [33]	The best competitive online algorithm for PAES

6.1 Data Traces and Comparison Algorithms

6.1.1 Data Center Energy Demands. For representing the energy demands of data centers, we use a dataset including the server load information for 300+ Akamai data centers across the United States, collected every 5 minutes [23]. Since some data centers are co-located with on-site renewable sources, we use wind data traces from [3] and inject renewable sources with 40% penetration in our experiments, unless the penetration level is otherwise stated. Two sample one week trajectories of energy demand for different locations in the United States are depicted in Figure 2. While we see a roughly diurnal pattern for the New York City energy demand, the pattern is less visible for Rochester. For both cities, the high unpredictability of renewable generation leads to comparable unpredictability in the net energy demand, regardless of diurnal patterns in energy demand. These observations show the importance of ML advice, as well as the possibility of tuning the level of trust in a principled manner.

6.1.2 Energy Pricing Data. We use the 2018 spot energy prices from the New York Electricity Market (NYISO). The value of spot prices changes in real-time over intervals of 5 minutes. As an example, the spot prices in April 2018 vary between \$13.69/MWh and \$64.62/MWh. The value of p_m is set to be roughly $100 \times \max_{t\in\mathcal{T}} p(t)$, which is based on common practice by U.S. utilities such as PG&E and Duke Energy. The cost of local generation is set to $p_g = \max_{t\in\mathcal{T}} p(t)$. Finally, the capacity of the local generator is set to be roughly 60% of the energy demand.

6.1.3 Comparison Algorithms. Table 2 summarizes the acronyms for different algorithms in our experiments. Here, we use two approaches to determine the trust parameter: *first*, offline optimal selection of the trust parameter – this approach searches over all possible values of λ in a brute force manner as input to OnMLEng and selects the best performing choice of λ. Although selecting the best choice of λ is not possible in online settings, the optimal hybrid algorithm serves to demonstrate the full potential of algorithms with ML advice. In experiments, the offline optimal algorithm OnMLEng-opt is used for PAES. *Second*, as a practical online selection, we choose the trust parameter based on the historical optimal value, that is the best λ for the previous instance of the problem. The algorithm OnMLEng-hist is used for this scenario. To demonstrate time-varying predictions, algorithm OnMLEng-hist-dyn aligns the predictions used in OnMLEng-hist according to observed values in an online manner.

In experiments, we report the normalized cost of different algorithms. The normalized cost is the ratio between the cost of the algorithm and the offline optimal cost (i.e., ENG-OPT for PAES). The normalized cost is always greater than or equal to 1. The lower the cost ratio of an algorithm, the better the performance. Finally, to show how online algorithms with ML advice achieve the best of both worlds, we compare their normalized cost to pure online algorithms (OnEng [33]) and naive data-driven algorithms that fully trust the advice (Eng-dd).

6.2 Large-scale Trace-Driven Evaluation

6.2.1 Analysis at Single Renewable Penetration Level. We first evaluate the performance of the proposed algorithms over a large variety of locations and trials, with emphasis on demonstrating how the proposed algorithms are able to achieve the best of both worlds. To begin, we focus on the cumulative probability distribution of normalized cost at a single penetration level. In Figure 3, we report results for PAES over 338 locations and 30 trials at 30% renewable penetration. Specifically, we observe that OnEng is strictly upper bounded by the theoretically guaranteed bound of approximately 1.85, but the majority (over 80%) of locations have normalized cost of greater than 1.6. The Eng-dd algorithm has comparatively better normalized cost for the majority of locations, but has a heavy tail. OnMLEng-opt and OnMLEng-hist clearly outperform OnEng and Eng-dd since they leverage advice for better decision making. Last but not least, the performance of OnMLEng and its variants largely depend on the level of uncertainty in energy demand. To investigate the impact of this uncertainty, we report additional experimental results at varying renewable penetration levels in Appendix F.

6.2.2 Evaluating a Dynamic Break-even Point Algorithm. A natural choice for a dynamic break-even point algorithm within OnMLEng-dyn is one that aligns the predictions with observed past and current values, i.e. once $p(t), e(t)$ are observed at time τ' then the predictions for all current and future time slots $\tau \geq \tau'$ are set as $\hat{p}_{\tau\geq\tau'}(t) = p(t), \hat{e}_{\tau\geq\tau'}(t) = e(t)$. This type of algorithm is a natural middle ground between OnEng and OnMLEng, since the predicted break-even points are gradually aligning with observed values. We consider a variant of this algorithm OnMLEng-hist-dyn which uses the best historical trust parameter, and compare it against OnMLEng-hist. From Figure 4, we see that OnMLEng-hist-dyn is slightly better than OnMLEng-hist in worst case and 99 percentile scenarios, but functionally equivalent in the average case scenario. This correlates closely with Theorem 4, as the two algorithms have equivalent theoretical robustness bounds.

6.3 Evaluation Results for PAES

In this section, we investigate the impact of different parameters on the performance of the proposed algorithms.

6.3.1 The Impact of Trust Parameter. Introducing the trust parameter in the algorithm design allows effective usage of predictions in algorithmic actions. Specifically, setting λ close to 0 represents more trust in predictions, while λ close to 1 represents almost no trust in predictions. To scrutinize the impact of λ on the performance of OnMLEng, in Figure 5(a) we vary the value of λ from 0 to 1. We report the average normalized cost over several locations

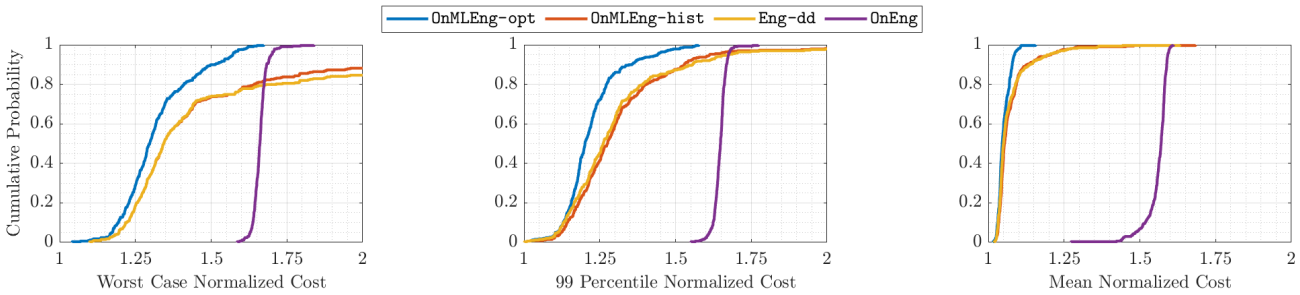


Figure 3: Cumulative probability distribution of normalized cost of different algorithms at 30% penetration level. We consider some key observations from these plots. First, algorithms with ML advice almost strictly outperform `OnEng` in mean normalized cost, but careful selection of trust level is important as prediction quality decreases. Second, the worst-case performance in `OnMLEng-hist` is noticeably robust to degrading prediction quality when compared to `Eng-dd`, indicating that careful selection of trust level will restrict poor worst-case performance.

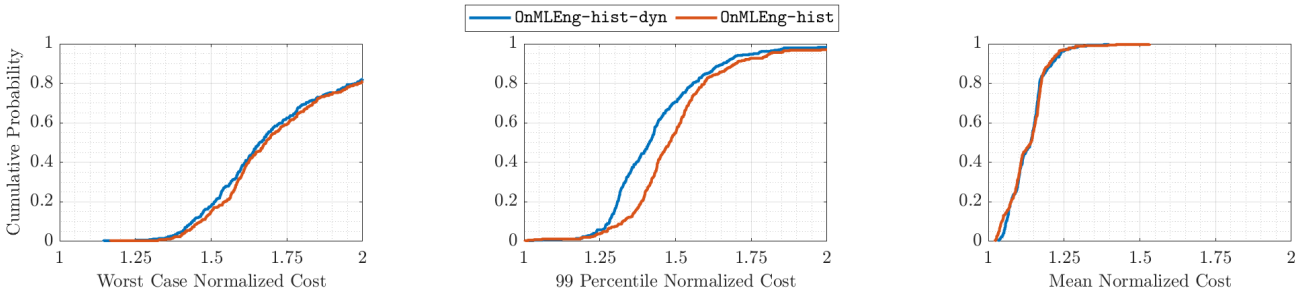
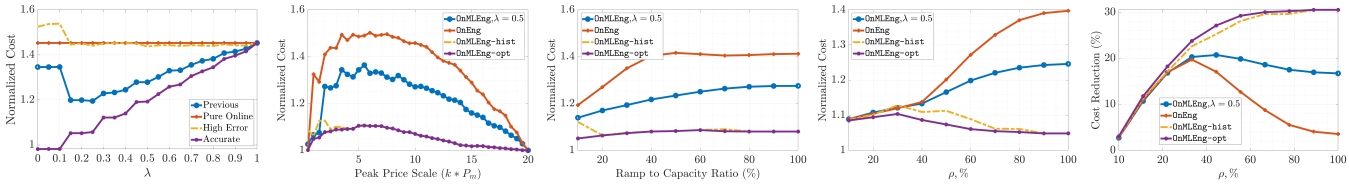


Figure 4: Cumulative probability distribution of dynamic vs. static break-even advice algorithms at 50% penetration. The key observation is that `OnMLEng-hist-dyn` is slightly better than `OnMLEng-hist` in worst case scenarios, but equivalent in the practical average case scenario.



(a) Trust parameter (b) Peak price (c) Ramp constraints (d) Capacity (e) Cost reduction

Figure 5: Evaluation results of five different experiments for PAES. Key observations are noted in 6.3.

and trials, with ML advice in three regimes: (i) *accurate* denotes perfect ML advice, (ii) *high error* denotes poor ML advice, and (iii) *previous* with the values of the previous run of the algorithm as the ML advice. These three regimes represent a broad range of ML advice and the goal is to investigate the impact of level of trust in different algorithms. The notable observations are summarized as follows: (1) With accurate ML advice, and $\lambda \leq 0.1$, `OnMLEng` achieves the optimal performance. (2) With high error in ML advice, unfavorable values of λ (high trust on prediction) lead to even worse performance than the pure online algorithms. (3) Favorable setting of λ, e.g., $\lambda \geq 0.4$ for `OnMLEng`, achieves better performance even with high error in ML advice. This experiment signifies the importance of setting right values for the trust parameter in order to outperform purely online algorithms without advice.

6.3.2 The Impact of Peak Price. The peak price p_m is an important parameter that impacts the break-even point. Case studies show that the peak charge varies substantially in different geographical locations, ranging from 20% to 80% of the total electricity bill [30]. In this experiment, we investigate the impact of this parameter on different algorithms. We scale the value of peak price from 1× to 20× of its original value and report the average normalized cost values of 30 runs in Figure 5(b). The result shows that the normalized cost of `OnMLEng` with trust $\lambda = 0.5$ is constantly better than `OnEng`. `OnMLEng-hist` is always very close to `OnMLEng-opt`

and is substantially better than OnEng. Interestingly, the normalized costs of all algorithms are better in the extremes of low and high peak prices. This is reasonable since with low peak prices it is natural to use the grid. At high peak prices, the optimal decision is clearly to fully utilize the generators. So, despite the uncertainty of the input, decisions in these two extreme regimes are trivial.

6.3.3 The Impact of Ramp Constraints. The algorithms proposed in this paper work for fast-response generators. In practice, there are several generators that are slow-response and cannot switch their output level quickly. The proposed algorithms are easily modified to incorporate ramp constraints. Specifically, let R be the ramp constraints, so that $|u(t) - u(t-1)| \leq R, \forall t \in \mathcal{T}$, i.e., the changes in generator output level should be always less than R. We can easily modify OnMLEng and OnEng, as explained in [34, Section 4], to reflect the ramp constraint. The idea is to first run the algorithm without the ramp constraints, and then, project the obtained values to the feasible region to respect the ramp constraints. In Figure 5(c), we vary the ramp to capacity ratio from 10% to 100%, and report the average normalized cost of OnEng and OnMLEng. The result shows that OnMLEng always achieves better performance than OnEng. Although the normalized costs for OnEng and OnMLEng increase as we relax the ramp constraints, for OnMLEng-hist and OnMLEng-opt those values are robust.

6.3.4 The Impact of Local Generation Capacity. A drawback of pure online algorithms such as OnEng is that they are too conservative in decision making. An example of such performance degradation is once the capacity of the generator is above 60% of the total energy demand (see Figure 5 in [33]). By leveraging ML advice, we can effectively prevent this performance degradation. To show this, we investigate the cost saving of different algorithms as the capacity of generator changes. We define $\rho = C/\max_t e(t)$ as the ratio between the capacity of generator and the maximum energy demand, and change this value from 10% to 100%. Figure 5(d) shows the normalized cost of of different algorithms. To better illustrate the benefit of algorithms with ML advice, in Figure 5(e) we report the cost reductions as compared to a baseline without local generation. With $\rho \leq 30\%$, all algorithms perform more or less similarly. However, with $\rho > 40\%$ the performance of OnEng and OnMLEng with $\lambda = 0.5$ degrades substantially, while the cost reduction of OnMLEng-opt and OnMLEng-hist increases. We consider this observation as another critical motivation to use online algorithms with ML advice for tackling online problems.

7 ALGORITHMIC DISCUSSION

The framework of ML advice for online algorithms is recently proposed and has been utilized for several online problems, e.g., online caching [22, 26, 28], bin packing [4], ski rental [5, 24, 27, 29] and job scheduling [24, 29]. However, this work is the first that uses this framework in the context of energy scheduling. Traditional approaches incorporating predictions often assume the prediction or prediction error follows a particular distribution or stochastic process, which limits the generality and practicality of their prediction framework. Other works [10, 11] use prediction windows of limited size that do not provide any information about events further into the future. These works apply predictions directly into the optimization for decision making. In this paper, predictions are used to generate advice for decision making. From our theoretical analysis and numerical evaluation, using advice is more powerful because only high-level structure such as break-even points is needed. This is in contrast to requiring detailed prediction of each time slot and modelling the error structure of each prediction.

PAES is an extended version of the ski-rental problem [8], in which a skier is going to ski for an unknown number of days. For each day, the skier can either rent skis at unit price or buy them for a higher price of $b > 1$ and ski for free from then on. The best known deterministic algorithm for ski-rental problem is the *break-even* algorithm: rent the first $b - 1$ days and buy on day b. In PAES, there is a *rent-vs-buy* dilemma in usage-based vs. peak-based decision making and the online algorithms in literature follow the break-even structure [33]. However, there is an additional unique challenge dedicated to PAES, namely that the "buy" option is not fully free and has an additional time-varying unit price. In our algorithm design with ML advice, we assume that an ML model provides an estimate of the break-even point to the problem. We do not assume any modeling from ML and treat it as a black-box that provides input to our algorithms.

8 CONCLUDING REMARKS

This paper improves the performance of classic competitive algorithms with ML advice in a principled manner for the peak-aware energy scheduling problem. Different from prior literature on using prediction for online algorithms, our algorithms are empowered with a parameter that determines the level of trust in the ML advice. For all algorithms we characterized the competitive ratio as a function of the trust parameter and showed that our algorithms are provably the best possible algorithms in this framework since they are Pareto optimal. By extensive large-scale experiments we showed the improved performance of the proposed algorithms against pure online algorithms as well as data-driven algorithms that naively trust the advice, verifying that our algorithms achieve the best of both worlds. While we focused on an energy scheduling problem, the rent-vs-buy nature and the category of break-even point algorithms appear frequently in broad application domains such as server on/off scheduling, TCP acknowledgment, and renting cloud servers, and a promising future direction is to extend the break-even point algorithms for those problems. Another promising direction is to incorporate the energy storage systems into the peak-aware energy scheduling problem.

ACKNOWLEDGMENTS

Russell Lee and Mohammad Hajiesmaili acknowledge the support from the U.S. National Science Foundation (NSF) under grant numbers CNS-1908298, NGSDI-2105494, and CAREER-2045641. Jessica Maghakian and Zhenhua Liu's research is supported in part by the NSF under grant numbers CNS-1717588, CNS-1730128, CNS-1919752, CNS-1839287, Graduate Research Fellowship, and an IBM Academic Award. Jian Li received support from the U.S. DOE Office of Energy Efficiency and Renewable Energy (EERE) under the Solar Energy Technologies Office Award Number DE-EE0009341. Also, Ramesh Sitaraman received support from the NSF under grant numbers CNS-1763617 and NGSDI-2105494.

REFERENCES

[1] 2018. Generator Permit Indicates Microsoft Plans Big Quincy Data Center Expansion. available at https://www.datacenterknowledge.com/microsoft/generator-permit-indicates-microsoft-plans-big-quincy-data-center-expansion, accessed May 2021.

[2] 2019. Global Data Center Generator Market to Generate Revenues of $5 Billion During 2018-2023. available at https://www.prnewswire.com/news-releases/global-data-center-generator-market-to-generate-revenues-of-5-billion-during-20182023--market-research-by-arizton-300789765.html, accessed May 2021.

[3] 2020. Eastern and Western Data Sets. available at https://www.nrel.gov/grid/eastern-western-wind-data.html.

[4] Spyros Angelopoulos, Christoph Dürr, Shendan Jin, Shahin Kamali, and Marc Renault. 2019. Online Computation with Untrusted Advice. *arXiv preprint arXiv:1905.05655* (2019).

[5] Etienne Bamas, Andreas Maggiori, and Ola Svensson. 2020. The Primal-Dual method for Learning Augmented Algorithms. In *Proc. of NeurIPS*.

[6] A. Borodin and R El-Yaniv. 1998. *Online computation and competitive analysis*. Cambridge University Press.

[7] Joan Boyar, Lene M Favrholdt, Christian Kudahl, Kim S Larsen, and Jesper W Mikkelsen. 2016. Online algorithms with advice: a survey. *Acm Sigact News* 47, 3 (2016), 93–129.

[8] Niv Buchbinder, Joseph Seffi Naor, et al. 2009. The design of competitive online algorithms via a primal–dual approach. *Foundations and Trends® in Theoretical Computer Science* 3, 2–3 (2009), 93–263.

[9] Niangjun Chen, Anish Agarwal, Adam Wierman, Siddharth Barman, and Lachlan LH Andrew. 2015. Online convex optimization using predictions. In *ACM SIGMETRICS Performance Evaluation Review*, Vol. 43. ACM, 191–204.

[10] Niangjun Chen, Joshua Comden, Zhenhua Liu, Anshul Gandhi, and Adam Wierman. 2016. Using predictions in online optimization: Looking forward with an eye on the past. *ACM SIGMETRICS Performance Evaluation Review* 44, 1 (2016), 193–206.

[11] Joshua Comden, Sijie Yao, Niangjun Chen, Haipeng Xing, and Zhenhua Liu. 2019. Online Optimization in Cloud Resource Provisioning: Predictions, Regrets, and Algorithms. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 3, 1 (2019), 16.

[12] Richard Evans and Jim Gao. 2016. DeepMind AI Reduces Google Data Centre Cooling Bill by 40%. *https://deepmind.com/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-40/* (2016).

[13] Íñigo Goiri, William Katsak, Kien Le, Thu D Nguyen, and Ricardo Bianchini. 2013. Parasol and greenswitch: Managing datacenters powered by renewable energy. In *ACM SIGPLAN Notices*, Vol. 48. ACM, 51–64.

[14] Sreenivas Gollapudi and Debmalya Panigrahi. 2019. Online Algorithms for Rent-Or-Buy with Expert Advice. In *International Conference on Machine Learning*. 2319–2327.

[15] Mohammad H Hajiesmaili, Chi-Kin Chau, Minghua Chen, and Longbo Huang. 2016. Online microgrid energy generation scheduling revisited: The benefits of randomization and interval prediction. In *In Proceedings of the ACM Seventh International Conference on Future Energy Systems*.

[16] Rohan Kodialam. 2019. Optimal Algorithms for Ski Rental with Soft Machine-Learned Predictions. *arXiv preprint arXiv:1903.00092* (2019).

[17] Tan N Le, Jie Liang, Zhenhua Liu, Ramesh K Sitaraman, Jayakrishnan Nair, and Bong Jun Choi. 2018. Optimal Energy Procurement for Geo-distributed Data Centers in Multi-timescale Electricity Markets. *ACM SIGMETRICS Performance Evaluation Review* 45, 2 (2018), 185–197.

[18] Minghong Lin, Adam Wierman, Lachlan LH Andrew, and Eno Thereska. 2013. Dynamic right-sizing for power-proportional data centers. *IEEE/ACM Transactions on Networking* 21, 5 (2013), 1378–1391.

[19] Zhenhua Liu, Yuan Chen, Cullen Bash, Adam Wierman, Daniel Gmach, Zhikui Wang, Manish Marwah, and Chris Hyser. 2012. Renewable and cooling aware workload management for sustainable data centers. In *ACM SIGMETRICS Performance Evaluation Review*, Vol. 40. 175–186.

[20] Zhenhua Liu, Minghong Lin, Adam Wierman, Steven Low, and Lachlan LH Andrew. 2014. Greening geographical load balancing. *IEEE/ACM Transactions on Networking* 23, 2 (2014), 657–671.

[21] Lian Lu, Jinlong Tu, Chi-Kin Chau, Minghua Chen, and Xiaojun Lin. 2013. Online energy generation scheduling for microgrids with intermittent energy sources and co-generation. *ACM SIGMETRICS Performance Evaluation Review* 41, 1 (2013), 53–66.

[22] Thodoris Lykouris and Sergei Vassilvtiskii. 2018. Competitive Caching with Machine Learned Advice. In *International Conference on Machine Learning*. 3302–3311.

[23] Erik Nygren, Ramesh K Sitaraman, and Jennifer Sun. 2010. The akamai network: a platform for high-performance internet applications. *ACM SIGOPS Operating Systems Review* 44, 3 (2010), 2–19.

[24] Manish Purohit, Zoya Svitkina, and Ravi Kumar. 2018. Improving Online Algorithms via ML Predictions. In *Advances in Neural Information Processing Systems*. 9661–9670.

[25] Asfandyar Qureshi, Rick Weber, Hari Balakrishnan, John Guttag, and Bruce Maggs. 2009. Cutting the Electric Bill for Internet-scale Systems. In *Proc. ACM SIGCOMM*.

[26] Dhruv Rohatgi. 2020. Near-Optimal Bounds for Online Caching with Machine Learned Advice. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 1834–1845.

[27] Shufan Wang, Jian Li, and Shiqiang Wang. 2020. Online Algorithms for Multi-shop Ski Rental with Machine Learned Advice. In *Proc. of NeurIPS*.

[28] Alexander Wei. 2020. Better and Simpler Learning-Augmented Online Caching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

[29] Alexander Wei and Fred Zhang. 2020. Optimal Robustness-Consistency Trade-offs for Learning-Augmented Online Algorithms. In *Proc. of NeurIPS*.

[30] Hong Xu and Baochun Li. 2014. Reducing Electricity Demand Charge for Data Centers with Partial Execution. In *Proc. ACM e-Energy*. 51–61.

[31] Lin Yang, Mohammad H Hajiesmaili, Ramesh Sitaraman, Enrique Mallada, Wing S Wong, and Adam Wierman. 2019. Online Inventory Management with Application to Energy Procurement in Data Centers. *arXiv preprint arXiv:1901.04372* (2019).

[32] Andrew Chi-Chin Yao. 1977. Probabilistic computations: Toward a unified measure of complexity. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*. IEEE, 222–227.

[33] Ying Zhang, Mohammad H Hajiesmaili, Sinan Cai, Minghua Chen, and Qi Zhu. 2018. Peak-aware online economic dispatching for microgrids. *IEEE Transactions on Smart Grid* 9, 1 (2018), 323–335.

[34] Ying Zhang, H Mohammad, Sinan Cai, Minghua Chen, and Qi Zhu. 2015. Peak-Aware Online Economic Dispatching for Microgrids. *arXiv preprint arXiv:1508.03996* (2015).

A PROOF OF ROBUSTNESS AND CONSISTENCY OF RONMLENG

A.1 Proof of Corollary 3 and Corollary 4

Proof. First we show the consistency and robustness bounds of Corollary 3 and Corollary 4 hold when $\hat{\sigma} > 1$.

(i) $\hat{\sigma} > 1, \sigma \leq \lambda < 1$. Note this is an incorrect prediction scenario, so the final upper bound lies in the robustness setting. The expected cost is given by $\int_s h(s,\sigma) f_1^*(s) ds$

$$
\begin{aligned}
&= \Phi_1 \int_0^{\sigma} \left[1 + \frac{1-\sigma+s}{\sigma}(1-\beta)\right] e^s ds + \Phi_1 \int_{\sigma}^{\lambda} e^s ds + \Phi_1(1)\lambda^2\beta \\
&= \Phi_1(e^{\lambda} - 1 + \lambda^2\beta) + \Phi_1(1-\beta) \\
&= 1 + \Phi_1(1-\beta).
\end{aligned}
$$

(ii) $\hat{\sigma} > 1, \lambda \leq \sigma < 1$. Note this is an incorrect prediction scenario, so the final upper bound lies in the robustness setting. The expected cost is given by $\int_s h(s,\sigma) f_1^*(s) ds$

$$
\begin{aligned}
&= \Phi_1 \int_0^{\lambda} \left[1 + \frac{1-\sigma+s}{\sigma}(1-\beta)\right] e^s ds + \Phi_1(1)\lambda^2\beta \\
&\leq \Phi_1 \int_0^{\lambda} e^s ds + \Phi_1 \int_0^{\sigma} \left[\frac{1-\sigma+s}{\sigma}(1-\beta)\right] e^s ds + \Phi_1\lambda^2\beta \\
&= 1 + \Phi_1(1-\beta).
\end{aligned}
$$

Case (ii) is clearly upper bounded by case (i), so the robustness bound holds.

(iii) $\hat{\sigma} > 1, \lambda < 1 < \sigma$. Note this is a best case correct prediction scenario, so the final upper bound lies in the consistency setting. The expected cost is given by $\int_s h(s,\sigma) f_1^*(s) ds$

$$
= \Phi_1 \int_0^{\lambda} \left[1 + \frac{s(1-\beta)}{(\sigma-1)\beta+1}\right] e^s ds + \Phi_1 \left[1 + \frac{(\sigma-1)(1-\beta)}{(\sigma-1)\beta+1}\right] \lambda^2\beta
$$

$$= 1 + (1-\beta)\Phi_1\left[\lambda^2 + \frac{(e^\lambda - 1 - \lambda)(\lambda - 1)}{(\sigma-1)\beta + 1}\right]$$
$$\overset{(a)}{\leq} 1 + (1-\beta)\Phi_1[\lambda^2 + 0] \leq 1 + \Phi_1\lambda^2(1-\beta),$$

where (a) holds true since we have $0 \leq e^\lambda - 1 - \lambda$ for $\lambda \in (0,1]$ from the discussions in Case (i), and $\lambda - 1 \leq 0$.

We now consider the cases where $\hat{\sigma} \leq 1$.

(iv) $\hat{\sigma} \leq 1, 1 \leq \frac{1}{\lambda} < \sigma$. Note this is a worst case failed prediction scenario. The expected cost is given by $\int_s h(s,\sigma) f_2^*(s) ds$

$$= \Phi_2 \int_0^{1/\lambda} \left[1 + \frac{s(1-\beta)}{(\sigma-1)\beta+1}\right] e^s ds + \Phi_2\left[1 + \frac{(\sigma-1)(1-\beta)}{(\sigma-1)\beta+1}\right]\frac{1}{\lambda^2}\beta$$
$$= 1 + \Phi_2(1-\beta)\left[\frac{e^{\frac{1}{\lambda}}(\frac{1}{\lambda}-1) + 1 + \frac{1}{\lambda^2}(\sigma-1)\beta}{(\sigma-1)\beta+1}\right]$$
$$= 1 + \Phi_2(1-\beta)\left[\frac{(e^{\frac{1}{\lambda}} - 1 - \frac{1}{\lambda})(\frac{1}{\lambda}-1)}{(\sigma-1)\beta+1} + \frac{1}{\lambda^2}\right]$$
$$\overset{(c)}{\leq} 1 + \Phi_2(1-\beta)\left[\frac{(e^{\frac{1}{\lambda}} - 1 - \frac{1}{\lambda})(\frac{1}{\lambda}-1)}{1} + \frac{1}{\lambda^2}\right],$$

where (c) holds since $(\sigma-1)\beta + 1 \geq 1$ and $(e^{\frac{1}{\lambda}} - 1 - \frac{1}{\lambda})$ is positive as shown in case (i) that $f(1/\lambda)$ is increasing for $\lambda \in [0,1]$ so $f(1/\lambda) \geq f(1) > 0$.

(v) $\hat{\sigma} \leq 1, 1 \leq \sigma < \frac{1}{\lambda}$. Note this is a worst case failed prediction scenario. The expected cost is given by $\int_s h(s,\sigma) f_2^*(s) ds$

$$= \Phi_2 \int_0^{\sigma} \left[1 + \frac{s(1-\beta)}{(\sigma-1)\beta+1}\right] e^s ds +$$
$$\Phi_2\left[1 + \frac{(\sigma-1)(1-\beta)}{(\sigma-1)\beta+1}\right]\left[\int_\sigma^{\frac{1}{\lambda}} e^s ds + \frac{1}{\lambda^2}\beta\right]$$
$$= 1 + \frac{\Phi_2(1-\beta)}{(\sigma-1)\beta+1}\left[1 + (\sigma-1)e^{\frac{1}{\lambda}} + (\sigma-1)\frac{1}{\lambda^2}\beta)\right]$$
$$\overset{(e)}{\leq} 1 + \frac{\Phi_2(1-\beta)}{(\sigma-1)\beta+1}\left[1 + (\frac{1}{\lambda}-1)e^{\frac{1}{\lambda}} + (\sigma-1)\frac{1}{\lambda^2}\beta)\right]$$
$$\leq 1 + \Phi_2(1-\beta)\left[(e^{\frac{1}{\lambda}} - 1 - \frac{1}{\lambda})(\frac{1}{\lambda}-1) + \frac{1}{\lambda^2}\right],$$

where (e) is true since $\sigma < \frac{1}{\lambda}$. The final step is is true since inequality (e) is equivalent to inequality (b) from case (iv), so the competitive ratio of case (v) reduces to the competitive ratio of case (iv). So the same robustness bound from (iv) will also dominate OnMLEng.

(vi) $\hat{\sigma} \leq 1, \sigma \leq 1 < \frac{1}{\lambda}$. This is a correct prediction scenario. The expected cost is given by $\int_s h(s,\sigma) f_2^*(s) ds$

$$= \Phi_2 \int_0^{\sigma} \left[1 + \frac{1-\sigma+s}{\sigma}(1-\beta)\right] e^s ds + \Phi_2 \int_\sigma^{\lambda} e^s ds + \Phi_2(1)\frac{1}{\lambda^2}\beta$$
$$= \Phi_2(e^\lambda - 1 + \frac{1}{\lambda^2}\beta) + \Phi_2(1-\beta)$$
$$= 1 + \Phi_2(1-\beta).$$

□

A.2 Proof of Corollary 5

Proof. We show that Corollary 5 is true for each of the above 6 cases. We begin with the cases where $\hat{\sigma} > 1$.

(1) Consider case (i) above with bound $1 + \Phi_1(1-\beta)$. We show that this bound is better that of the deterministic algorithm, i.e., $1 + \Phi_1(1-\beta) \leq 1 + \frac{1}{\lambda}(1-\beta)$ for $\lambda \in (0,1]$ and $\beta \in [0,1]$. In other words, we need to show $\Phi \leq \frac{1}{\lambda}$, i.e., $\frac{1}{e^\lambda - 1 + \lambda^2\beta} \leq \frac{1}{\lambda}$. Since $\frac{1}{e^\lambda - 1 + \lambda^2\beta} \leq \frac{1}{e^\lambda - 1}$, we only need to prove that $\frac{1}{e^\lambda - 1} \leq \frac{1}{\lambda}$, i.e., $e^\lambda - 1 \geq \lambda$ for $\lambda \in (0,1]$. Define $f(\lambda) = e^\lambda - 1 - \lambda$. It is easy to check that $f(\lambda)$ is increasing in $(0,1]$, hence $f(\lambda) \geq f(0) = 0$.

(2) The result for case (i) holds, since case (ii) is upper bounded by case (i).

(3) Consider case (iii) above with bound $1 + \lambda^2\Phi_1(1-\beta)$. We show that this bound is better than that of the deterministic algorithm, i.e., $1 + \lambda^2\Phi_1(1-\beta) \leq 1 + \lambda(1-\beta)$ for $\lambda \in (0,1]$ and $\beta \in [0,1]$. Given Φ_1 we only need to show $\frac{\lambda^2}{e^\lambda - 1 + \lambda^2\beta} \leq \lambda$, i.e., $\frac{1}{e^\lambda - 1 + \lambda^2\beta} \leq \frac{1}{\lambda}$, which holds true from case (i).

So cases (1) - (3) demonstrate that when $\hat{\sigma} > 1$, rOnMLEng dominates the robustness and consistency bounds of OnMLEng.

New, we consider the cases where $\hat{\sigma} \leq 1$. (4) Consider case (iv) above with bound $1 + \Phi_2(1-\beta)((e^{\frac{1}{\lambda}} - 1 - \frac{1}{\lambda})(\frac{1}{\lambda} - 1) + \frac{1}{\lambda^2})$. We show that this bound is better than that of OnMLEng, i.e., $1 + \Phi_2(1-\beta)((e^{\frac{1}{\lambda}} - 1 - \frac{1}{\lambda})(\frac{1}{\lambda} - 1) + \frac{1}{\lambda^2}) \leq 1 + \frac{1}{\lambda}(1-\beta)$, i.e., $\Phi_2((e^{\frac{1}{\lambda}} - 1 - \frac{1}{\lambda})(\frac{1}{\lambda} - 1) + \frac{1}{\lambda^2}) \leq \frac{1}{\lambda}$. To show this, we have

$$\Phi_2((e^{\frac{1}{\lambda}} - 1 - \frac{1}{\lambda})(\frac{1}{\lambda} - 1) + \frac{1}{\lambda^2}) \leq \frac{1}{\lambda}$$
$$\Leftrightarrow (e^{\frac{1}{\lambda}} - 1 - \frac{1}{\lambda})(\frac{1}{\lambda} - 1) + \frac{1}{\lambda^2} \leq \frac{1}{\lambda\Phi_2}$$
$$\Leftrightarrow (e^{\frac{1}{\lambda}} - 1 - \frac{1}{\lambda})(\frac{1}{\lambda} - 1) + \frac{1}{\lambda^2} \leq \frac{1}{\lambda}(e^{\frac{1}{\lambda}} - 1 + \frac{1}{\lambda^2}\beta)$$
$$\Leftrightarrow \frac{1}{\lambda}(e^{\frac{1}{\lambda}} - 1 - \frac{1}{\lambda}) + \frac{1}{\lambda^2} - (e^{\frac{1}{\lambda}} - 1 - \frac{1}{\lambda}) \leq \frac{1}{\lambda}(e^{\frac{1}{\lambda}} - 1 + \frac{1}{\lambda^2}\beta)$$
$$\Leftrightarrow (\frac{1}{\lambda}e^{\frac{1}{\lambda}} - \frac{1}{\lambda} - \frac{1}{\lambda^2} + \frac{1}{\lambda^2}) - (e^{\frac{1}{\lambda}} - 1 - \frac{1}{\lambda}) \leq (\frac{1}{\lambda}e^{\frac{1}{\lambda}} - \frac{1}{\lambda} + \frac{1}{\lambda^3}\beta)$$
$$\Leftrightarrow (\frac{1}{\lambda}e^{\frac{1}{\lambda}} - \frac{1}{\lambda}) - (e^{\frac{1}{\lambda}} - 1 - \frac{1}{\lambda}) \leq (\frac{1}{\lambda}e^{\frac{1}{\lambda}} - \frac{1}{\lambda} + \frac{1}{\lambda^3}\beta)$$
$$\Leftrightarrow 0 - (e^{\frac{1}{\lambda}} - 1 - \frac{1}{\lambda}) \leq 0 + \frac{1}{\lambda^3}\beta$$
$$\Leftrightarrow 0 - (e^{\frac{1}{\lambda}} - 1 - \frac{1}{\lambda}) \overset{(d)}{\leq} 0 \leq 0 + \frac{1}{\lambda^3}\beta,$$

where (d) holds true from $(e^{\frac{1}{\lambda}} - 1 - \frac{1}{\lambda})$ is positive. The inequality holds true, thus the robustness bound of rOnMLEng dominates OnMLEng when $\hat{\sigma} \leq 1$.

(5) The result for case (iv) holds, since case (v) is upper bounded by case (iv).

(6) Consider case (vi) above with bound $1 + \Phi_2(1-\beta)$. We show that this bound is better than that of OnMLEng, i.e., $1 + \Phi_2(1-\beta) \leq 1 + \lambda(1-\beta)$, i.e., $\Phi_2 \leq \lambda$. It is easy to show that $e^{\frac{1}{\lambda}} - 1 \geq \frac{1}{\lambda}$. Then $e^{\frac{1}{\lambda}} - 1 + \frac{1}{\lambda^2}\beta \geq \frac{1}{\lambda}$, i.e., $\frac{1}{\Phi_2} \geq \frac{1}{\lambda}$ from the definition of Φ_2. Therefore, we have $\Phi_2 \leq \lambda$. Thus the competitive ratio dominates the consistency bound of OnMLEng. So cases (3) - (6) demonstrate that the rOnMLEng dominates the robustness and consistency bounds of OnMLEng when $\hat{\sigma} \leq 1$. □

B A RANDOMIZED ALGORITHM WITH DIRECT EXTENSION OF THE EXISTING RANDOMIZED ALGORITHM

The goal in this section is to show that a naive incorporation of the ML advice in designing a randomized algorithm lead to an algorithm that is neither robust nor consistent. Specifically, we show that a randomized algorithm that modifies the distribution function proposed in Equation (2) fails to achieve both robustness and consistency at the same time. In particular, a first attempt to change the distribution function is to naturally modify according to the enhancements in deterministic algorithms and obtain the following functions:

if $\hat{\sigma} > 1$:

$$f_1^*(s) = \begin{cases} \Phi_1 e^s, & s \in [0, \lambda]; \\ \Phi_1 \lambda \beta \delta(0) & s = \infty; \\ 0, & \text{o.w.}, \end{cases}$$

if $\hat{\sigma} \leq 1$:

$$f_1^*(s) = \begin{cases} \frac{\lambda e^s}{e^{1/\lambda}-1+\beta}, & s \in [0, 1/\lambda]; \\ \frac{\beta}{e^{1/\lambda}-1+\beta}\delta(0), & s = \infty; \\ 0, & \text{o.w.} \end{cases}$$

Our analysis below demonstrates that with these functions, rOnMLEng is $\max\left\{\min\left\{1/\beta, 1/\lambda\right\} \cdot \frac{e^{1/\lambda}}{e^{1/\lambda}-1+\beta}, \frac{e^{\lambda}}{e^{\lambda}-1+\beta}\right\}$-robust and $(1/\beta)$-consistent. This means that with above distribution functions the consistency could be large as β approaches 0.

(i) $\hat{\sigma} > 1, \sigma \leq \lambda < 1$. This is an incorrect prediction scenario. The expected cost is given by $\int_s h(s,\sigma) f_1^*(s) ds$

$$\begin{aligned} &= \int_0^{\sigma} \left[1 + \frac{1-\sigma+s}{\sigma}(1-\beta)\right] \frac{e^s}{e^{\lambda}-1+\beta} ds + \\ &\quad \int_{\sigma}^{\lambda} \frac{e^s}{e^{\lambda}-1+\beta} ds + (1)\frac{\beta}{e^{\lambda}-1+\beta} \\ &= \frac{e^{\lambda}-1}{e^{\lambda}-1+\beta} + \frac{1-\beta}{e^{\lambda}-1+\beta} + \frac{\beta}{e^{\lambda}-1+\beta} \\ &= \frac{e^{\lambda}}{e^{\lambda}-1+\beta}. \end{aligned}$$

(ii) $\hat{\sigma} > 1, \lambda \leq \sigma < 1$. This is an incorrect prediction scenario. The expected cost is given by $\int_s h(s,\sigma) f_1^*(s) ds$

$$\begin{aligned} &= \int_0^{\lambda} \left[1 + \frac{1-\sigma+s}{\sigma}(1-\beta)\right] \frac{e^s}{e^{\lambda}-1+\beta} ds + (1)\frac{\beta}{e^{\lambda}-1+\beta} \\ &\leq \int_0^{\lambda} \frac{e^s}{e^{\lambda}-1+\beta} ds + \int_0^{\sigma} \left[\frac{1-\sigma+s}{\sigma}(1-\beta)\right] \frac{e^s}{e^{\lambda}-1+\beta} ds + \\ &\quad \frac{\beta}{e^{\lambda}-1+\beta} \\ &= \frac{e^{\lambda}-1}{e^{\lambda}-1+\beta} + \frac{1-\beta}{e^{\lambda}-1+\beta} + \frac{\beta}{e^{\lambda}-1+\beta} \\ &= \frac{e^{\lambda}}{e^{\lambda}-1+\beta}. \end{aligned}$$

(iii) $\hat{\sigma} > 1, \lambda < 1 < \sigma$. Note this is a correct prediction scenario. The expected cost is given by $\int_s h(s,\sigma) f_1^*(s) ds$

$$\begin{aligned} &= \int_0^{\lambda} \left[1 + \frac{s(1-\beta)}{(\sigma-1)\beta+1}\right] \frac{e^s}{e^{\lambda}-1+\beta} ds + \\ &\quad \left[1 + \frac{(\sigma-1)(1-\beta)}{(\sigma-1)\beta+1}\right] \frac{\beta}{e^{\lambda}-1+\beta} \\ &= 1 + \frac{1}{e^{\lambda}-1+\beta}\left[\frac{(1-\beta)e^{\lambda}(\lambda-1)}{(\sigma-1)\beta+1} + (1-\beta)\right] \\ &= \frac{e^{\lambda}}{e^{\lambda}-1+\beta} - \frac{e^{\lambda}}{e^{\lambda}-1+\beta}\left[\frac{(1-\lambda)(1-\beta)}{(\sigma-1)\beta+1}\right] \\ &\leq \frac{e^{\lambda}}{e^{\lambda}-1+\beta}. \end{aligned}$$

(iv) $\hat{\sigma} \leq 1, 1 \leq 1/\lambda < \sigma$. This is an incorrect prediction scenario.The expected cost is given by $\int_s h(s,\sigma) f_2^*(s) ds$

$$\begin{aligned} &= \int_0^{1/\lambda} \left[1 + \frac{s(1-\beta)}{(\sigma-1)\beta+1}\right] \frac{e^s}{e^{1/\lambda}-1+\beta} ds + \\ &\quad \left[1 + \frac{(\sigma-1)(1-\beta)}{(\sigma-1)\beta+1}\right] \frac{\beta}{e^{1/\lambda}-1+\beta} \\ &= \frac{e^{1/\lambda}}{e^{1/\lambda}-1+\beta}\left[1 + \frac{(1/\lambda-1)(1-\beta)}{(\sigma-1)\beta+1}\right] \\ &\overset{(c)}{\leq} \frac{e^{1/\lambda}}{e^{1/\lambda}-1+\beta}\left[1 + (1/\lambda-1)(1-\beta)\right] \\ &\overset{(d)}{\leq} \frac{e^{1/\lambda}}{e^{1/\lambda}-1+\beta}\left[1 + (1/\lambda-1)\right] \\ &= \frac{1}{\lambda}\frac{e^{1/\lambda}}{e^{1/\lambda}-1+\beta}, \end{aligned}$$

where (c) holds since $(\sigma-1)\beta+1 \geq 1$, and (d) is true since $0 \leq 1-\beta \leq 1$. However, note the following upper bound also holds:

$$\begin{aligned} 1 + \frac{(1/\lambda-1)(1-\beta)}{(\sigma-1)\beta+1} &\leq 1 + \frac{(\sigma-1)(1-\beta)}{(\sigma-1)\beta+1} \\ &\leq 1 + \frac{(\sigma-1)(1-\beta)}{(\sigma-1)\beta} \\ &\leq 1 + \frac{(1-\beta)}{\beta} \\ &\leq \frac{1}{\beta}. \end{aligned}$$

Then the competitive ratio in this case is

$$\int_s h(s,\sigma) f_2^*(s) ds \leq \min\left\{1/\beta, 1/\lambda\right\} \cdot \frac{e^{1/\lambda}}{e^{1/\lambda}-1+\beta}.$$

(v) $\hat{\sigma} \le 1, 1 \le \sigma < 1/\lambda$. This is an incorrect prediction scenario. The expected cost is given by $\int_s h(s,\sigma) f_2^*(s) ds$

$$
\begin{aligned}
&= \int_0^{\sigma} \left[1 + \frac{s(1-\beta)}{(\sigma-1)\beta+1}\right] \frac{e^s}{e^{1/\lambda}-1+\beta} ds \\
&+ \left[1 + \frac{(\sigma-1)(1-\beta)}{(\sigma-1)\beta+1}\right] \left[\int_{\sigma}^{1/\lambda} \frac{e^s}{e^{1/\lambda}-1+\beta} ds + \frac{\beta}{e^{1/\lambda}-1+\beta}\right] \\
&= \frac{e^{1/\lambda}}{e^{1/\lambda}-1+\beta}\left[1 + \frac{(1/\lambda-1)(1-\beta)}{(\sigma-1)\beta+1}\right] \\
&= \frac{e^{1/\lambda}}{e^{1/\lambda}-1+\beta} \frac{(\sigma-1/\lambda)\beta + \frac{1}{\lambda}}{(\sigma-1)\beta+1} \\
&\overset{(d)}{\le} \frac{e^{1/\lambda}}{e^{1/\lambda}-1+\beta} \frac{1/\lambda}{1} = \frac{1}{\lambda} \frac{e^{1/\lambda}}{e^{1/\lambda}-1+\beta},
\end{aligned}
$$

where (d) is true since $1 \le \sigma \le 1/\lambda$, $\sigma - 1/\lambda < 0$, and $\sigma - 1 \ge 0$. Then the competitive ratio in this case is

$$
\int_s h(s,\sigma) f_2^*(s) ds \le \min\left\{1/\beta, 1/\lambda\right\} \cdot \frac{e^{1/\lambda}}{e^{1/\lambda}-1+\beta}.
$$

(vi) $\hat{\sigma} \le 1, \sigma \le 1 < 1/\lambda$. Note this is a correct prediction scenario. The expected cost is given by $\int_s h(s,\sigma) f_2^*(s) ds$

$$
\begin{aligned}
&= \int_0^{\sigma} \left[1 + \frac{1-\sigma+s}{\sigma}(1-\beta)\right] \frac{e^s}{e^{1/\lambda}-1+\beta} ds + \\
&\int_{\sigma}^{1/\lambda} \frac{e^s}{e^{1/\lambda}-1+\beta} ds(1) \frac{\beta}{e^{1/\lambda}-1+\beta} \\
&= \int_0^{1/\lambda} \frac{e^s}{e^{1/\lambda}-1+\beta} ds + \int_0^{\sigma} \left[\frac{1-\sigma+s}{\sigma}(1-\beta)\right] \cdot \frac{e^s}{e^{1/\lambda}-1+\beta} ds \\
&+ \frac{\beta}{e^{1/\lambda}-1+\beta} \\
&= \frac{e^{1/\lambda}-1}{e^{1/\lambda}-1+\beta} + \frac{1-\beta}{e^{1/\lambda}-1+\beta} + \frac{\beta}{e^{1/\lambda}-1+\beta} = \frac{e^{1/\lambda}}{e^{1/\lambda}-1+\beta}.
\end{aligned}
$$

Next, we consider the consistency. For consistency guarantees, we compute the competitive ratio assuming the predictions are correct. There are two cases to consider here

(i) $\hat{\sigma} = \sigma > 1$. With a selected parameter s from the distribution $f_1^*(s)$, the algorithm uses the local generator for the first T^s time slots before switching to the grid. Then the cost of the algorithm is $\texttt{ALG} = \sum_{t=1}^{T^s} p_g e(t) + \sum_{t=T^s}^{T} p(t)e(t) + p_m$. Since $\sigma > 1$, the optimal offline solution uses the grid for the whole duration with cost $\texttt{OPT} = \sum_{t=1}^{T} p(t)e(t) + p_m$. Then we have the following:

$$
\begin{aligned}
\texttt{ALG} &= \sum_{t=1}^{T^s} p_g e(t) + \sum_{t=T^s+1}^{T} p(t)e(t) + p_m \\
&= \sum_{t=1}^{T^s} (p_g - p(t))e(t) + \sum_{t=1}^{T} p(t)e(t) + p_m \\
&\le s \cdot p_m + \sum_{t=1}^{T} p(t)e(t) + p_m \le (1+s) \cdot p_m + \sum_{t=1}^{T} p(t)e(t) \\
&\le (1+s)(p_m + \sum_{t=1}^{T} p(t)e(t)) \le (1+s)\texttt{OPT}.
\end{aligned}
$$

To compute the expected expected cost of the randomized algorithm, we need to know a special case of the cost of ALG when $s = \infty$. With $s = \infty$, the algorithm never switches to grid electricity

$$
\begin{aligned}
\texttt{ALG}_{\{s=\infty\}} &= \sum_{t=1}^{T} p_g e(t) = \sum_{t=1}^{T} \frac{p_g}{p(t)} p(t)e(t) \le \frac{p_g}{p^{min}} \sum_{t=1}^{T} p(t)e(t) \\
&= \frac{1}{\beta} \sum_{t=1}^{T} p(t)e(t) \le \frac{1}{\beta} \sum_{t=1}^{T} p(t)e(t) + \frac{1}{\beta} p_m = \frac{1}{\beta}\texttt{OPT}.
\end{aligned}
$$

Then the expected cost of the randomized algorithm is

$$
\begin{aligned}
\mathbb{E}[\texttt{ALG}] &= \int_s \texttt{ALG} \cdot f_1^*(s) ds \\
&\le \int_0^{\lambda} (1+s)(\texttt{OPT}) \frac{e^s}{e^{\lambda}-1+\beta} ds + \frac{1}{\beta}(\texttt{OPT}) \frac{\beta}{e^{\lambda}-1+\beta} \\
&\le \frac{\texttt{OPT}}{e^{\lambda}-1+\beta}\left[1 + \int_0^{\lambda} e^s + s e^s ds\right] \\
&= \frac{\texttt{OPT}}{e^{\lambda}-1+\beta}(1 + \lambda e^{\lambda}).
\end{aligned}
$$

If $\lambda = 0$, we have $(1/\beta)$-consist.

(ii) $\hat{\sigma} = \sigma \le 1$. With the trust parameter λ, the algorithm uses the local generator for the first $T^{1/\lambda}$ time slots before switching to the grid, where $T^{1/\lambda} \le T$. Then the cost of the algorithm is $\texttt{ALG} = \sum_{t=1}^{T^{1/\lambda}} p_g e(t) + \sum_{t=T^{1/\lambda}+1}^{T} p(t)e(t) + p_m$. Since $\sigma \le 1$, the optimal offline solution uses the grid for the whole duration with cost $\texttt{OPT} = \sum_{t=1}^{T} p_g e(t)$. Then we have the following:

$$
\begin{aligned}
\texttt{ALG} &= \sum_{t=1}^{T^{1/\lambda}} p_g e(t) + \sum_{t=T^{1/\lambda}+1}^{T} p(t)e(t) + p_m \\
&\le \sum_{t=1}^{T^{1/\lambda}} p_g e(t) + \sum_{t=T^{1/\lambda}+1}^{T} p_g e(t) + p_m = \texttt{OPT} + p_m \\
&\overset{(e)}{\le} \texttt{OPT} + \lambda\left(\sum_{t=1}^{T^{1/\lambda}+1} (p_g - p(t))e(t)\right) \\
&\le \texttt{OPT} + \lambda\left(\sum_{t=1}^{T} p_g e(t)\right) \\
&= (1+\lambda)\texttt{OPT},
\end{aligned}
$$

where (e) is true from Algorithm rOnMLEng.

C THE ROBUSTNESS AND CONSISTENCY OF ONMLENG-DYN AND RONMLENG-DYN

This appendix presents Algorithm 4 and Algorithm 5 as the pseudocode of the dynamic break-even algorithms OnMLEng-dyn and rOnMLEng-dyn introduced in Section 4.5.

C.1 Proof of Theorem 4

Proof. The key observation is that Proposition 1 still holds with a dynamic break-even point, where we can characterize the competitive ratio of any online algorithm $\mathcal{A}_s$ with parameter s.

Algorithm 4 `OnMLEng-dyn`

Use local generator first and switch to the grid starting at the first time τ where

$$\sum_{t=1}^{\tau}(p_g - p(t))e(t) \geq s_\tau \cdot p_m,$$

and s_τ is defined by

$$s_\tau = \begin{cases} \lambda, & \hat{\sigma}_t > 1; \\ 1/\lambda, & \hat{\sigma}_t \leq 1. \end{cases}$$

Algorithm 5 `rOnMLEng-dyn`

Denote $\Phi_1 = \frac{1}{e^{\lambda}-1+\lambda^2\beta}$ and $\Phi_2 = \frac{1}{e^{\frac{1}{\lambda}}-1+\frac{1}{\lambda^2}\beta}$

if $\hat{\sigma}_\tau > 1$ **then**

$$f_1^*(s) = \begin{cases} \Phi_1 e^s, & s \in [0, \lambda]; \\ \Phi_1\lambda^2\beta\delta(0), & s = \infty; \\ 0, & \text{otherwise.} \end{cases}$$

else

$$f_2^*(s) = \begin{cases} \Phi_2 e^s, & s \in [0, \frac{1}{\lambda}]; \\ \Phi_2 \frac{1}{\lambda^2}\beta\delta(0), & s = \infty; \\ 0, & \text{otherwise.} \end{cases}$$

end if

Pick a value s_1 randomly according to probability distribution $f_1^*(s)$ and s_2 likewise from $f_2^*(s)$.

Switch to grid electricity starting at the first time τ where

$$\sum_{t=1}^{\tau}(p_g - p(t))e(t) \geq s_\tau \cdot p_m,$$

and s_τ is defined by

$$s_\tau = \begin{cases} s_1, & \hat{\sigma}_t > 1; \\ s_2, & \hat{\sigma}_t \leq 1. \end{cases}$$

Although the value of s will change over time in `OnMLEng-dyn` with dynamic advice $\hat{\sigma}_t$, the possible values of s remain the same as the possible values in `OnMLEng`. The possible competitive ratios must be the same, and subsequently the robustness and consistency are the same. Similarly for `rOnMLEng-dyn` and `rOnMLEng`, s may be time varying but is still constrained to the same two distributions.

□

D PROOF OF LEMMA 2

We show that an arbitrary deterministic algorithm can be expressed by a deterministic algorithm with a switching parameter.

We first define two algorithms: (i) `Generic-set-selection`, a deterministic algorithm that is not limited by a switching parameter; and (ii) `Converted-switching-parameter`, rearranges price and demand $p(t), e(t)$ such that `Generic-set-selection` is replicated.

Algorithm 6 `Generic-set-selection`

Use local generator for a set of timeslots $T_l \subseteq \mathcal{T}$, and use grid electricity starting for the set of timeslots $T_g = \mathcal{T} - T_l$

Algorithm 7 `Converted-switching-parameter`

Let the timeslots of T_l be specified $T_l = \{l_1, l_2, ..l_{|T_l|}\}$ and T_g be specified $T_g = \{g_1, g_2, ..g_{|T_g|}\}$.

Define a new ordering of price and demand $p'(t), e'(t)$ according to:

$$p'(t) = \begin{cases} p(l_t), & \text{if } t \leq |T_l|, \\ p(g_t), & \text{otherwise.} \end{cases}$$

$$e'(t) = \begin{cases} e(l_t), & \text{if } t \leq |T_l|, \\ e(g_t), & \text{otherwise.} \end{cases}$$

Choose switching parameter s according to:

$$\frac{1}{p_m}\sum_{t=1}^{|T_l|}(p_g - p'(t))e'(t) = s.$$

Under the new ordering, use local generator first and switch to the grid electricity starting at the first time τ where

$$\sum_{t=1}^{\tau}(p_g - p'(t))e'(t) \geq s \cdot p_m.$$

We consider Algorithm 6 with T_l, T_g, and an arbitrary deterministic algorithm $\mathcal{A}$ modeled by Algorithm 6 with T_l, T_g.

Let t^* be the last timeslot before $|T_l|$ in the new demand ordering $e'(t)$ with nonzero demand, i.e. t^* is defined by

$$t^* = \max_{t \leq |T_l|, e'(t)=1} t.$$

We will show that the switching parameter algorithm will switch at time t^*. In other words, the timeslots chosen for local generator and the grid will be the same as T_l and T_g, except for some timeslots from t^* to $|T_l|$ with 0 demand. As a results, the cost is equivalent since timeslots with 0 demand contribute nothing to the cost.

Since $e'(t) = 0$ for $t^* < t \leq |T_l|$, we have

$$\begin{aligned} s &= \frac{1}{p_m}\sum_{t=1}^{|T_l|}(p_g - p'(t))e'(t) \\ &= \frac{1}{p_m}\sum_{t=1}^{t^*}(p_g - p'(t))e'(t) + \frac{1}{p_m}\sum_{t=t^*+1}^{|T_l|}(p_g - p'(t)) \cdot 0 \\ &= \frac{1}{p_m}\sum_{t=1}^{t^*}(p_g - p'(t))e'(t) + 0, \end{aligned}$$

i.e., $\sum_{t=1}^{t^*}(p_g - p'(t))e'(t) = s \cdot p_m$.

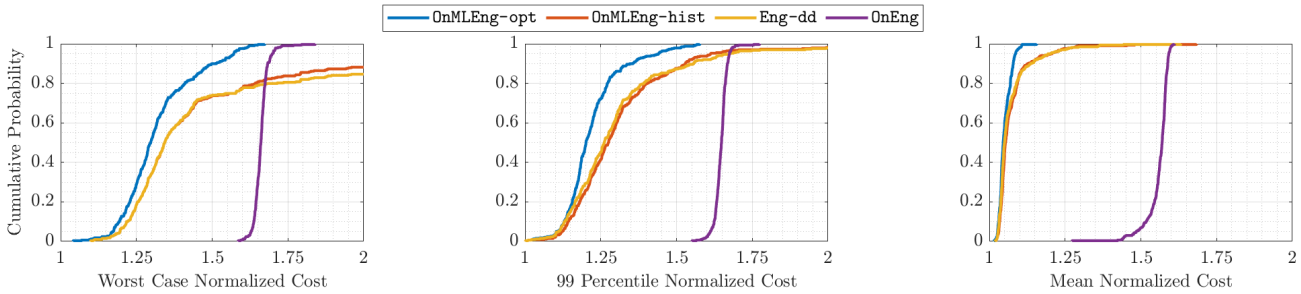
Similarly, since $e'(t^*) = 1, p'(t) < p_g, \forall t$, we have

$$\begin{aligned} s &= \frac{1}{p_m}\sum_{t=1}^{t^*}(p_g - p'(t))e'(t) \\ &= \frac{1}{p_m}\sum_{t=1}^{t^*-1}(p_g - p'(t))e'(t) + \frac{(p_g - p'(t^*))e'(t^*)}{p_m} \\ &> \frac{1}{p_m}\sum_{t=1}^{t^*-1}(p_g - p'(t))e'(t), \end{aligned}$$

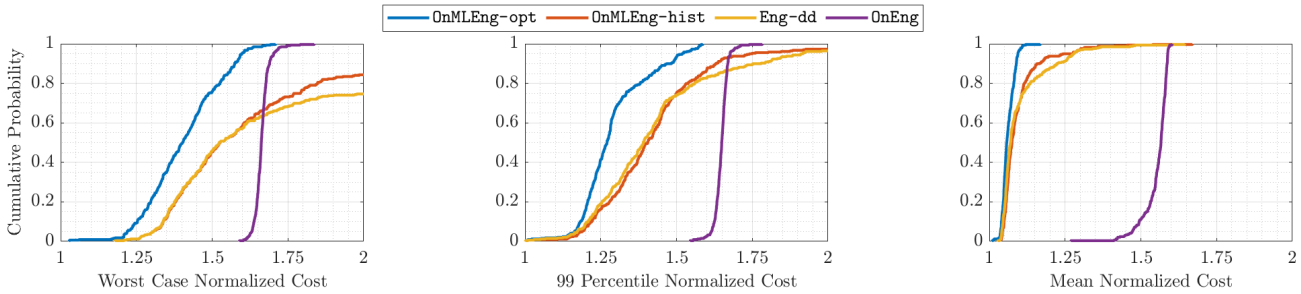
i.e., $\sum_{t=1}^{t^*-1}(p_g - p'(t))e'(t) < s \cdot p_m$.

Therefore the first time τ where $\sum_{t=1}^{\tau}(p_g - p'(t))e'(t) \geq s \cdot p_m$ will be at $\tau = t^*$.

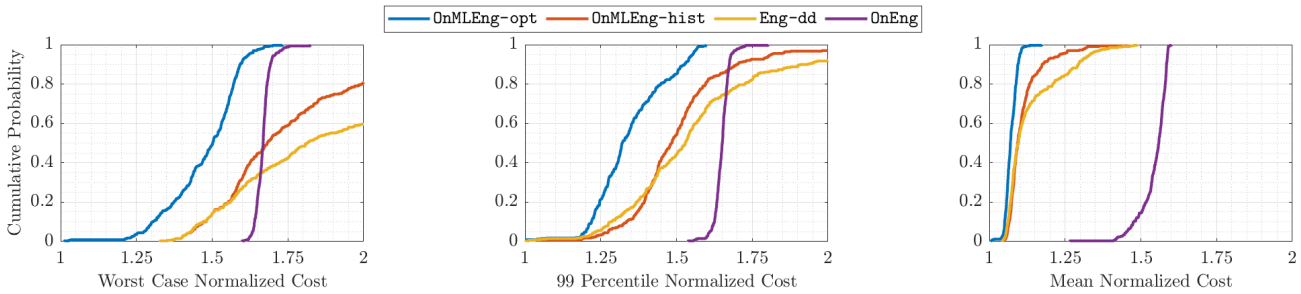
Timeslots $1, \cdots, t^*$ are selected for the local generator. These are correctly assigned since $t^* \leq |T_l|$. If $t^* < |T_l|$, then timeslots $t^* < t \leq |T_l|$ are incorrectly assigned to the grid. However, $e'(t) = 0$ for $t^* < t \leq |T_l|$, which means there is no difference in cost. Timeslots



(a) 30% penetration.



(b) 40% penetration.



(c) 50% penetration.

Figure 6: Cumulative probability distribution of normalized cost of different algorithms at 30%, 40%, and 50% penetration levels.

$|T_l|+1, \cdots, T$ are correctly assigned to the grid. Therefore switching at time t^* in the new ordering has an equivalent cost as assigning T_l and T_g.

E EXTENDING RESULTS TO THE GENERAL DEMAND SETTING

Consider an instance of PAES with integer demand. We can construct a binary demand instance PAES-B at the k-th layer by denoting $e^k(t) = \mathbf{1}_{\{e(t)\geq k\}}$. The full details of decomposing PAES into PAES-B are inspired by [33], so we outline the necessary adaptions for robustness and consistency analysis.

Denote $v^k(t)$ and $u^k(t)$ the energy usage from the grid and local generator respectively from the k-th layer of binary demand. Note that $\max_t \sum_k v^k(t) = \sum_k \max_t v^k(t)$, i.e. the overall peak grid utilization is the sum of the layered peak utilization. Similarly, $u(t) = \sum_k u^k(t), v(t) = \sum_k v^k(t)$, i.e. the overall grid and generator usage is the sum of the layered grid and generator usage. Then we have

$$\texttt{cost}(\text{PAES} - \texttt{ALG}) = \sum_k \texttt{cost}(\text{PAES-B} - \texttt{ALG})$$

i.e. the cost of PAES is equal to the sum of the costs over the binary demand problems PAES-B.

E.1 Extending Consistency and Robustness Results

Let an algorithm which solves PAES-B be α-robust and γ-consistent in the binary demand setting. Then we demonstrate that extending to the integer demand setting PAES is also α robust and γ consistent. Consider the consistency scenario, where the integer demand predictions $\hat{e}(t)$ are correct. Then each layer prediction $\hat{e}^k(t)$would

also be correct. We can then use the γ consistency bound. Consider a binary demand layer k:

$$\texttt{cost}(\text{PAES-B} - \texttt{ALG}) \leq \gamma \texttt{cost}(\text{PAES-B} - \texttt{OPT}), \forall k$$

Then summing over k gives:

$$\texttt{cost}(\text{PAES} - \texttt{ALG}) \leq \gamma \texttt{cost}(\text{PAES} - \texttt{OPT})$$

Now consider the robustness scenario, where the overall demand prediction $\hat{e}(t)$ is not necessarily accurate. Then each consider a binary demand layer k will be α robust:

$$\texttt{cost}(\text{PAES-B} - \texttt{ALG}) \leq \alpha \texttt{cost}(\text{PAES-B} - \texttt{OPT}), \forall k$$

Then summing over k gives:

$$\texttt{cost}(\text{PAES} - \texttt{ALG}) \leq \alpha \texttt{cost}(\text{PAES} - \texttt{OPT})$$

We can just substitute the respective consistency and robustness bounds of `OnMLEng` and `rOnMLEng` for γ and α. Thus the upper bounds on `OnMLEng` and `rOnMLEng` extend to the general demand setting.

E.2 Extending the Pareto Optimality of `OnMLEng`

Observe that PAES-B is a special case of PAES. Then it is impossible for an algorithm which solves PAES to dominate `OnMLEng` in the integer demand setting. If such an algorithm existed, then it would dominate `OnMLEng` in the binary demand setting, which contradicts Theorem 2. Thus the pareto optimality of `OnMLEng` extends to the general demand setting.

F ADDITIONAL EXPERIMENTAL RESULTS WITH DIFFERENT RENEWABLE PENETRATION

The quality of advice for PAES could be substantially influenced by the renewable penetration level. Hence, it is also valuable to see a comparison of algorithms at varying penetration levels. The experiments in Figure 3 were done at 30% penetration, but experiments at 40% (Figure 6(b)) and 50% (Figure 6(c)) are also shown for comparison. For better illustration of the results, the results for 40% penetration are also included again in Figure 6(b). Overall, increasing the penetration degrades the accuracy of the predictions and subsequently drives the normalized cost higher. This is most prevalent in the worst case scenario, where the heavy tails beyond the theoretical guarantee increase from 20% to 40% for `Eng-dd`. On the other hand, the mean normalized cost is relatively robust degrading predictions via penetration level. For all algorithms, the mean normalized cost remains below the theoretical guarantee.