Learning-Augmented Dynamic Power Management with Multiple States via New Ski Rental Bounds

Anonymous Author(s)
Affiliation
Address
email

Abstract

We study the online problem of minimizing power consumption in systems with multiple power-saving states. During idle periods of unknown lengths, an algorithm has to choose between power-saving states of different energy consumption and wake-up costs. We develop a learning-augmented online algorithm that makes decisions based on (potentially inaccurate) predicted lengths of the idle periods. The algorithm's performance is near-optimal when predictions are accurate and degrades gracefully with increasing prediction error, with a worst-case guarantee almost identical to the optimal classical online algorithm for the problem. A key ingredient in our approach is a new algorithm for the online ski-rental problem in the learning augmented setting with tight dependence on the prediction error. We support our theoretical findings with experiments.

1 Introduction

Energy represents up to 70% of total operating costs of modern data centers [36] and one of the major quality-of-service parameters in battery-operated devices. In order to ameliorate this, contemporary CPUs are equipped with sleep states to which the processor can transition during periods of inactivity. In particular, the ACPI-standard [22] specifies that each processor should possess, along with the active state $C0$ that is used for processing tasks, at least one sleep state $C1$. Modern processors generally possess more sleep states $C2, \ldots$; for example, current Intel CPUs implement at least 4 such C-states [17]. Apart from CPUs, such sleep states appear in many systems ranging from hard drives or mobile devices to the start-stop feature found in many cars, and are furthermore often employed when rightsizing data centers [2].

Intuitively, in a "deeper" sleep state, the set of switched-off components will be a superset of the corresponding set in a more shallow sleep state. This implies that the *running cost* for residing in that deeper state will be lower, but the *wake-up* cost to return to the active state $C0$ will be higher compared to a more shallow sleep state. In other words, there is a tradeoff between the running and the wake-up cost. During each idle period, a *dynamic power management (DPM)* strategy has to decide in which state the system resides at each point in time, without a-priori knowledge about the duration of the idle period. Optimally managing these sleep states is a challenging problem due to its online nature. On the one hand, transitioning the system to a too deep state could be highly suboptimal if the idle period ends shortly after. On the other hand, spending too much idle time in a shallow state would accumulate high running costs.

The special case of 2-state DPM systems, i.e., when there is only a single sleep state (besides the active state), is essentially equivalent to the *ski rental* problem, one of the most classical problems and of central importance in the area of online optimization [34; 23]. This problem is defined as follows: A person goes skiing for an unknown number of days. On every day of skiing, the person must decide whether to continue renting skis for one more day or to buy skis. Once skis are bought

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

there will be no more cost on the following days, but the cost of buying is much higher than the cost of renting for a day. It is easy to see that this captures a single idle period of DPM with a single sleep state whose running cost is 0: The rental cost corresponds to the running cost of the active state and the cost of buying skis corresponds to the wake-up cost; transitioning to the sleep state corresponds to buying skis. Given this equivalence, the known 2-competitive deterministic algorithm and $e/(e-1) \approx 1.58$-competitive randomized algorithm for ski-rental carry over to 2-state DPM, and these competitive ratios are tight.

Irani et al. [25] extended the deterministic algorithm for ski rental/2-state DPM to the case of multiple states, still achieving a competitive ratio of 2. They also gave a deterministic $e/(e-1)$-competitive algorithm for the case in which the length of the idle periods is repeatedly drawn from a fixed, and known, probability distribution. When the probability distribution is fixed but unknown they developed an algorithm that *learns* the distribution over time and showed that it performs well in practice. Although it is perhaps not always reasonable to assume a fixed underlying probability distribution for the length of idle periods, real-life systems do often follow periodical patterns so that these lengths can indeed be frequently predicted with adequate accuracy, see Chung et al. [16] for a specific example. Nevertheless, it is not hard to see that blindly following such predictions can lead to arbitrarily bad performance when predictions are faulty. The field of *learning-augmented* algorithms [33] is concerned with algorithms that incorporate predictions in a robust way.

In this work, we introduce multi-state DPM to the learning-augmented setting. Inspired by the 2-competitive deterministic algorithm of Irani et al. [25], we give a formal reduction from multi-state DPM to the ski rental problem, which works also for randomized algorithms and even in the learning-augmented setting. Although ski-rental has been investigated through the learning-augmented algorithms lens before [35; 39], earlier work has focused on the optimal trade-off between *consistency* (i.e., the performance when predictions are accurate) and *robustness* (i.e., the worst-case performance). To apply our reduction from DPM to ski rental, we require more refined guarantees for learning-augmented ski rental. To this end we develop a new learning-augmented algorithm for ski-rental that obtains the optimal trade-off between consistency and dependence on the prediction error. Our resulting algorithm for DPM achieves a competitive ratio arbitrarily close to 1 in case of perfect predictions and its performance degrades gracefully to a competitive ratio arbitrarily close to the optimal robustness of $e/(e-1) \approx 1.58$ as the prediction error increases.

Potential negative societal impact. This is a work of theoretical nature and we are not aware of potential negative societal impact. That said, we cannot rule out future misuse of the contained theoretical knowledge.

1.1 Formal definitions

Problem definition. In the problem of *dynamic power management (DPM)*, we are given $k+1$ power states denoted by $0, 1, \ldots, k$, with power consumptions $\alpha_0 > \cdots > \alpha_k \geq 0$ and wake-up costs $\beta_0 < \cdots < \beta_k$. For state 0, we have $\beta_0 = 0$ and we call this the *active state*. The input is a series of *idle periods* of lengths $\ell_1, \ldots, \ell_T$ received online, i.e., the algorithm does not know the length of the current period before it ends. During each period, the algorithm can transition to states with lower and lower power consumption, paying energy cost $x\alpha_i$ for residing in state i for time x. If j is the state at the end of the idle period, then it has to pay the wake-up cost β_j to transition back to the active state 0. The goal is to minimize the total cost.

In the *learning-augmented* setting, the algorithm receives at the beginning of ith idle period a prediction $\tau_i \geq 0$ for the value of ℓ_i as additional input. We define $\eta_i := \alpha_0|\tau_i - \ell_i|$ to be the error of the ith prediction, and $\eta := \sum_i^T \eta_i$ to be the total prediction error.

(Continuous-time) *ski-rental* is the special case of DPM with $k = 1$, $\alpha_1 = 0$ and a single idle period of some length ℓ. In this case, we call $\alpha := \alpha_0$ the rental cost, $\beta := \beta_1$ the buying cost, and ℓ the length of the ski season. In learning-augmented ski rental, we write the single prediction as $\tau := \tau_1$.

(ρ, μ)-competitiveness. Classical online algorithms are typically analyzed in terms of *competitive ratio*. An algorithm $\mathcal{A}$ for an online minimization problem is said to be *ρ-competitive* (or alternatively, obtain a *competitive ratio* of ρ) if for any input instance,

$$cost(\mathcal{A}) \leq \rho \cdot \textsc{Opt} + c,$$

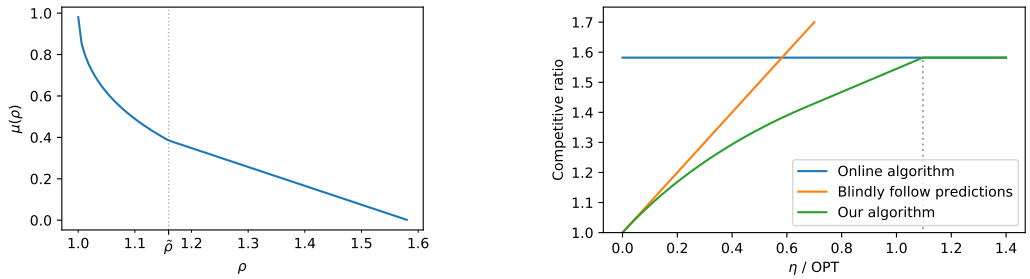

Figure 1: Illustration of $\mu(\rho)$ and of the resulting competitive ratio in function of $\eta/\textsc{Opt}$.

where $cost(\mathcal{A})$ and $\textsc{Opt}$ denote the cost of $\mathcal{A}$ and the optimal cost of the instance and c is a constant independent of the *online* part of the input (i.e., the lengths ℓ_i in case of DPM). For the ski rental problem one requires $c = 0$, since the trivial algorithm that buys at time 0 has constant cost β.

In the learning-augmented setting, for $\rho \geq 1$ and $\mu \geq 0$, we say that $\mathcal{A}$ is (ρ, μ)-*competitive* if

$$cost(\mathcal{A}) \leq \rho \cdot \textsc{Opt} + \mu \cdot \eta \tag{1}$$

for any instance, where η is the prediction error. This corresponds to a competitive ratio of $\rho + \mu \frac{\eta}{\textsc{Opt}}$ (with $c = 0$). While this could be unbounded as $\eta/OPT \to \infty$, our DPM algorithm achieves a favorable competitive ratio even in this case (see Theorem 5, where we take the minimum over a range of pairs (ρ, μ), including $\mu = 0$).

For a (ρ, μ)-competitive algorithm, ρ is also called the *consistency* (i.e., competitive ratio in case of perfect predictions) while μ describes the dependence on the prediction error.

1.2 Our results

Our first result is a (ρ, μ)-*competitive algorithm* for ski rental that achieves the optimal μ corresponding to the given ρ. For $\rho \in [1, \frac{e}{e-1}]$, let

$$\mu(\rho) := \max\left\{\frac{1 - \rho\frac{e-1}{e}}{\ln 2},\ \rho(1-T)e^{-T}\right\}, \tag{2}$$

where $T \in [0, \frac{1}{2}]$ is the solution to $T^2 e^{-T} = 1 - \frac{1}{\rho}$. Let $\tilde{\rho} \approx 1.16$ be the value of ρ for which both terms in the maximum yield the same value. The first term dominates for $\rho > \tilde{\rho}$ and the second term if $\rho < \tilde{\rho}$. Note that $\mu(1) = 1$ and $\mu\left(\frac{e}{e-1}\right) = 0$. See Figure 1 (left) for an illustration.

Theorem 1. *For any $\rho \in [1, \frac{e}{e-1}]$, there is a $(\rho, \mu(\rho))$-competitive randomized algorithm for learning-augmented ski rental, i.e., given a prediction with error η, its expected cost is at most $\rho\,\textsc{Opt} + \mu(\rho) \cdot \eta$.*

Note that $\rho < 1$ is impossible for any algorithm (due to the case $\eta = 0$) and $\rho > \frac{e}{e-1}$ is uninteresting since $\rho = \frac{e}{e-1}$ already achieves the best possible value of $\mu = 0$.

We also prove a lower bound showing that $\mu(\rho)$ defined in (2) is the best possible.

Theorem 2. *For any $\rho \in [1, \frac{e}{e-1}]$ and any (randomized) algorithm $\mathcal{A}$, there is a ski rental instance with some prediction error η such that the expected cost of $\mathcal{A}$ is at least $\rho\,\textsc{Opt} + \mu(\rho)\eta$.*

However, for most values of the prediction τ it is possible to achieve a better $\mu < \mu(\rho)$, and $\mu(\rho)$ only captures the worst case over all possible predictions τ. In the supplementary material, we describe how to achieve the best possible μ as a function of both ρ and τ. The proof of Theorem 1 is sketched in Section 2, with details as well as the proof of Theorem 2 deferred to the supplementary material.

In Section 3, we give a reduction from DPM to ski rental provided that the ski rental algorithm satisfies a natural monotonicity property (defined formally in Section 3):

Theorem 3. *If there is a monotone (ρ, μ)-competitive ski rental algorithm, then there is a (ρ, μ)-competitive algorithm for DPM.*

Since our ski rental algorithm is monotone, this directly yields a $(\rho, \mu(\rho))$-competitive algorithm for DPM. From the special case $(\rho, \mu) = \left(\frac{e}{e-1}, 0\right)$, we also obtain the following result for classical DPM (without predictions), which to the best of our knowledge has not been reported before:

Corollary 4. *There is a $\frac{e}{e-1}$-competitive randomized online algorithm for DPM (without predictions).*

Using techniques from online learning, in a way similar to [5], we show in Section 4 how to achieve "almost" $(\rho, \mu(\rho))$-competitiveness simultaneously for *all* ρ:

Theorem 5. *For any $\epsilon > 0$, there is a learning-augmented algorithm $\mathcal{A}$ for dynamic power management whose expected cost can be bounded as*

$$cost(\mathcal{A}) \leq (1+\epsilon) \min \left\{ \rho \, \textsc{Opt} + \mu(\rho) \cdot \eta \;\middle|\; \rho \in [1, \tfrac{e}{e-1}] \right\} + O\left(\tfrac{\beta_k}{\epsilon} \log \tfrac{1}{\epsilon}\right).$$

The above theorem gives a competitive ratio arbitrarily close to $\min\{\rho + \mu(\rho) \cdot \frac{\eta}{\textsc{Opt}}\}$, which is equal to 1 if $\eta = 0$ and never greater than $\frac{e}{e-1}$. In particular, we achieve a performance that degrades gracefully from near-optimal consistency to near-optimal robustness as the error increases.[1] See Figure 1 (right) for an illustration.

In Section 5, we illustrate the performance of these algorithms by simulations on synthetic datasets, where the dependence on the prediction error can be observed as expected from theoretical results.

1.3 Related work

Learning-Augmented Algorithms: Learning augmented algorithms have been a very active area of research since the seminal paper of Lykouris and Vassilvitskii [32]. We direct the interested reader to a survey [33] by Mitzenmacher and Vassilvitskii, as well as [7; 18; 32; 5; 37; 38; 31; 29] for recent results on secretary problems, paging, k-server as well as scheduling problems. In the following we survey some results in the area more closely related to our work.

The ski-rental problem has already been studied within the context of learning augmented algorithms. Here, the main objective was to optimize the tradeoff between consistency and robustness (performance on perfect predictions and worst-case performance). The first results were due to Purohit et al. [35] who propose a deterministic and a randomized algorithm. They also present a linear dependency on the error: their randomized algorithm is (ρ, ρ)-competitive for $\rho \geq 1$, with larger ρ allowing for better robustness. A hyperparameter allows to choose a prescribed consistency and leads to a corresponding robustness. Wei and Zhang [39] show that the consistency / robustness tradeoff achieved by the randomized algorithm of [35] is Pareto-optimal. Angelopoulos et al. [4] propose a deterministic algorithm achieving a Pareto-optimal consistency / robustness tradeoff, but with no additional guarantee when the error is small. A variant with multiple predictions was studied in [19].

As we will see in Section 4, DPM can be cast as a problem from the class of *Metrical Task Systems (MTS)*. Antoniadis et al. [5] gave a learning-augmented algorithm for MTS that can be interpreted as $(1, 4)$-competitive within their prediction setup.

A different problem related to energy conservation is the classical online speed scaling problem, which was recently studied in the learning-augmented setting by Bamas et al. [9].

DPM: The equivalence between 2-state DPM and ski-rental is mentioned in [34]. Therefore the well-known 2-competitive deterministic and an $e/(e-1)$-competitive randomized algorithm [27] for the classical ski-rental problem carry over to 2-state DPM, and these bounds are known to be tight.

Irani et al. [25] present an extension of the 2-competitive algorithm for two-state DPM to multi-state DPM that also achieves a competitive ratio of 2. Furthermore they give an $e/(e-1)$-competitive algorithm for the case that the lengths of the idle periods come from a fixed probability distribution.

There have been several previous approaches that try to predict the length of an idle interval (see, e.g., [16; 25], and the survey of Benini et al. [11]). However, the proposed approaches to use these predictions are not robust against a potentially high prediction error.

Augustine et al. [8] investigate a problem generalizing DPM where transition cost is paid for going to a deeper sleep state rather than waking up and these transition costs may be non-additive (i.e., it can be cheaper to skip states). Albers [2] studies the offline version of the problem with multiple, parallel devices and shows that it can be solved in polynomial time.

[1] At first glance, our consistency and robustness might seem to contradict the lower bound of Wei and Zhang [39] for ski rental. However, [39] crucially uses $c = 0$ in the definition of competitiveness for ski rental.

Irani et al. [26] introduced a 2-state problem where jobs that need to be processed have a release-time, a deadline and a required processing time. This gives further flexibility to the system to schedule the jobs and create periods of inactivity so as to maximize the energy-savings by transitioning to the sleep state. For the offline version, there is an exact polynomial-time algorithm due to Baptiste et al. [10]. Recently, a 3-approximation algorithm for the multiprocessor-case was developed [6].

Another related problem consists of deciding which components of a data-center should be powered on or off in order to process the current load on the set of active components (see, e.g., [3]). A similar problem, where jobs have individual processing times for each machine, was studied in [28; 30].

Several surveys cover DPM, see for example [11; 1; 24].

2 New algorithm for ski rental

Throughout this section, let $\rho \in [1, e/(e-1)]$ be fixed and let $\mu = \mu(\rho)$ be as defined in Equation (2). We will present a (ρ, μ)-competitive algorithm for (learning augmented) ski-rental, proving Theorem 1. The next lemma shows that it suffices to give such an algorithm for $\alpha = \beta = 1$.

Lemma 6. *An algorithm $\mathcal{A}'$ that is (ρ, μ)-competitive for instances of the ski-rental problem with $\alpha = \beta = 1$ implies a (ρ, μ)-competitive algorithm $\mathcal{A}$ for arbitrary $\alpha, \beta > 0$.*

Proof idea. We simulate algorithm $\mathcal{A}$' with prediction $\frac{\alpha}{\beta}\tau$. If it buys at time t', then $\mathcal{A}$ buys at time $t = \frac{\beta}{\alpha}t'$. □

2.1 Description of the algorithm

We next describe a randomized algorithm for instances with $\alpha = \beta = 1$, which can then be used to solve arbitrary ski-rental instances using Lemma 6. Our algorithm is fully specified by the cumulative distribution function (CDF) F_τ of the time when the algorithm buys skis. The algorithm then draws a $p \in [0, 1]$ uniformly at random and buys at the earliest time $t \in [0, \infty)$ such that $F_\tau(t) \geq p$. The CDF F_τ will depend on the given prediction $\tau \geq 0$ as well as the fixed ρ and $\mu = \mu(\rho)$.

Definition of the CDF (see Figure 2) We denote by P_0 the probability of buying at time 0 and, for any $t > 0$, we let p_t be the probability density of buying at time t, so that the probability that the algorithm has bought until time x can be expressed as

$$F_\tau(x) = P_0 + \int_0^x p_t dt.$$

For convenience, we also specify the probability $P_\infty = 1 - (P_0 + \int_0^\infty p_t dt)$ of never buying.

To define P_0 and p_t, we distinguish three cases depending on the value of the prediction τ. Note that we always have $0 \leq \mu \leq 1 \leq \rho \leq \frac{e}{e-1}$.

Case 1: $\mu\tau \leq \mu - \rho + 1$. We choose

$$P_0 = \frac{\tau(\rho - 1)}{1 - \tau}, \qquad p_t = \begin{cases} \rho e^{t-1} & \text{for } t \in (b, 1] \\ 0 & \text{otherwise} \end{cases}, \qquad P_\infty = \min\{\mu, 1 - P_0\},$$

where $b \in [\tau, 1]$ is chosen such that $P_0 + P_\infty + \int_b^1 \rho e^{t-1} dt = 1$, in order to have the sum of probabilities equal to 1. Note that if $P_0 \geq 1 - \mu$, we have $b = 1$ and $p_t = 0$ for all $t > 0$.

Case 2: $\mu - \rho + 1 \leq \mu\tau \leq \mu$. We choose

$$P_0 = \mu\tau, \qquad p_t = \begin{cases} (\mu\tau + \rho - \mu - 1)e^t & \text{for } t \leq a \\ \rho e^{t-1} & \text{for } t \in (b, 1] \\ 0 & \text{otherwise} \end{cases}, \qquad P_\infty = \min\{\mu, 1 - P_0\},$$

where $a \in [0, \tau]$ is chosen maximal such that $P_0 + P_\infty + \int_0^a (\mu\tau + \rho - \mu - 1)e^t dt \leq 1$, and $b \in [\tau, 1]$ is chosen so that $P_0 + P_\infty + \int_0^a (\mu\tau + \rho - \mu - 1)e^t dt + \int_b^1 \rho e^{t-1} dt = 1$ in order to have the sum of probabilities equal to 1. In case $\rho = \frac{e}{e-1}$, we have $\mu = 0$ and $(\mu\tau + \rho - \mu - 1)e^t = (\rho - 1)e^t = \rho e^{t-1}$, recovering the classical online algorithm of Karlin et al. [27].

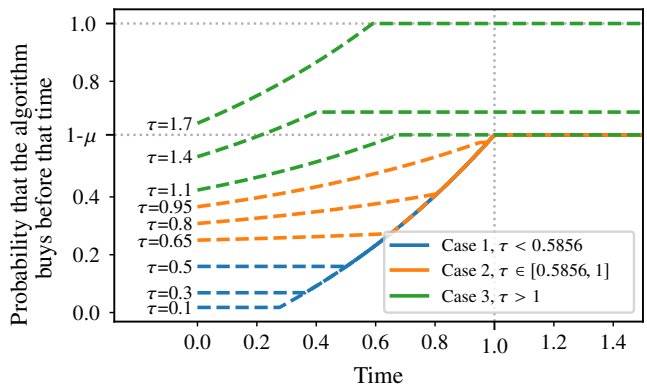

Figure 2: Our (ρ, μ)-competitive ski-rental algorithm for $\rho = \tilde{\rho} \approx 1.1596$ and $\mu = \mu(\tilde{\rho}) \approx 0.3852$. The figure presents the cummulative distribution functions of the time of buying for several prediction values τ. Here $\alpha = \beta = 1$, i.e., at time $t = 1$ buying and renting has equal costs.

Case 3: $\tau > 1$**.** If $\mu\tau \geq 1$, we buy at time 0. Otherwise, we choose

$$P_0 = \mu\tau, \qquad p_t = \begin{cases} (\mu\tau + \rho - \mu - 1)e^t & \text{if } t \leq T \\ 0 & \text{if } t > T \end{cases}, \qquad P_\infty = \rho - \mu - (\mu\tau + \rho - \mu - 1)e^T,$$

where T is the number closest to $\tau - 1$ that satisfies

$$e^T \leq \frac{\rho - \mu}{\mu\tau + \rho - \mu - 1} \qquad \text{(equivalently } P_\infty \geq 0) \tag{3}$$

$$e^T \geq \frac{\rho - 2\mu}{\mu\tau + \rho - \mu - 1} \qquad \text{(equivalently } P_\infty \leq \mu). \tag{4}$$

Thus, either $T = \tau - 1$ if this choice satisfies both bounds, or T is at an endpoint of the feasible interval prescribed by (3) and (4).

2.2 Sketch of the analysis

Our algorithm is (ρ, μ)-competitive if and only if for any $x \geq 0$ we have

$$cost(x) := P_0 + \int_0^x (1 + t)\, p_t\, dt + \int_x^\infty x\, p_t\, dt + xP_\infty \leq \rho \min\{x, 1\} + \mu|\tau - x|, \tag{5}$$

where $cost(x)$ denotes the expected cost of the algorithm in the case when $\ell = x$: If we intend to buy at some time t and $t < x$, we pay $1 + t$, otherwise we pay x. On the right hand side, $\min\{x, 1\}$ is the optimal cost and $|\tau - x|$ is the prediction error, assuming $\alpha = \beta = 1$.

We first sketch the analysis for Case 2, and then discuss the differences in Case 1. These cases are relatively simple. Case 3 is far more involved and we will only sketch the ideas.

Case 2: The inequality $\mu(\rho) \geq \frac{1-\rho\frac{e-1}{e}}{\ln 2}$ from the definition of $\mu(\rho)$ guarantees that $b \in [\tau, 1]$ exists. The worst case occurs only for $\tau = \ln 2$, otherwise one can also achieve a smaller μ. We give details in the supplementary material. We now show that (5) is satisfied.

Note that (5) is tight for $x = 0$, with both sides equal to $\mu\tau$. To obtain (5) for all $x > 0$, it suffices to show that the derivative of the left-hand side with respect to x is at most the derivative of the right-hand side (where derivatives exist). For $x \in (0, \infty) \setminus \{a, b, 1\}$, we have

$$\frac{d}{dx} cost(x) = (1 + x)p_x + \int_x^\infty p_t\, dt - xp_x + P_\infty = p_x + \int_x^\infty p_t\, dt + P_\infty.$$

For $x \in (0, a)$ this yields

$$\frac{d}{dx} cost(x) = p_x + 1 - P_0 - (p_x - p_0) = 1 - \mu\tau + (\mu\tau + \rho - \mu - 1)e^0 = \rho - \mu,$$

which is equal to the derivative of the right-hand side of (5). For $x \in (a, b)$, $\frac{d}{dx} cost(x)$ is even smaller because p_x is 0, and the derivative of the right-hand side of (5) is $\rho - \mu$ or $\rho + \mu$. For $x \in (b, 1)$,

$$\frac{d}{dx} cost(x) = p_x + \int_x^\infty p_t dt + P_\infty = p_x + (p_1 - p_x) + P_\infty = \rho + P_\infty \leq \rho + \mu,$$

which is equal to the derivative of the right-hand side of (5). Finally, for $x > 1$ we have $\frac{d}{dx} cost(x) = P_\infty \leq \mu$ and the derivative of the right-hand side is also μ.

Case 1: The reason we cannot define p_t in the same way as in Case 2 is that p_t would be negative for $t \leq a$ (i.e., the algorithm would try to sell skis that it bought at time 0, which is not allowed). We therefore choose P_0 such that (5) is tight for $x = \tau$ if we do not buy in the interval $(0, \tau]$. The remainder of the proof of (5) is similar to Case 2. The existence of $b \in [\tau, 1]$ follows from the inequality $\mu \geq \rho(1 - T)e^{-T}$ from the definition of $\mu(\rho)$, with the worst case occurring for $\tau = 1 - T$.

Case 3: The first step in the analysis of Case 3 is to derive an inequality involving ρ, μ, τ and T that is equivalent to the algorithm being (ρ, μ)-competitive. Denoting by $\mu_\tau(\rho)$ the minimal μ satisfying this inequality, it suffices to show that $\mu_\tau(\rho) \leq \mu(\rho)$ for all $\tau > 1$. However, the difficulty is that no closed-form expression for $\mu_\tau(\rho)$ exists. However, we are still able to show that $\tau \mapsto \mu_\tau(\rho)$ can have a local maximum only if $T = \tau - 1$, and therefore $\sup_{\tau > 1} \mu_\tau(\rho)$ is achieved either for $\tau \to 1$ or when $T = \tau - 1$. This allows us to eliminate τ from the aforementioned inequality, and we can then show that $\mu = \mu(\rho)$ satisfies the remaining inequality (with tightness occurring for $\rho \leq \tilde{\rho}$ and $\tau = T + 1$).

A full proof of Theorem 1 is provided in the supplementary material.

3 Reduction from DPM to ski rental

We now give a reduction from DPM to ski rental (Theorem 3), provided that the ski rental algorithm satisfies the following monotonicity property: We say that a ski rental algorithm for rental cost $\alpha = 1$ and buying cost $\beta = 1$ is *monotone* if its CDF F_τ for the buying time when given prediction τ satisfies

$$F_\tau(t) \leq F_{\tau'}(t) \qquad \text{for all } t \geq 0 \text{ and } \tau < \tau'.$$

Intuitively, this property is very natural: The longer the predicted duration of skiing, the greater should be our probability of buying. Indeed, our algorithm satisfies this property:

Lemma 7. *The ski rental algorithm from Section 2 is monotone.*

As mentioned earlier, for many τ one could actually achieve a better $\mu_\tau(\rho) < \mu(\rho)$. However, somewhat surprisingly the optimal such algorithm would *not* be monotone. The monotonicity of our algorithm therefore crucially relies on our specific description (in particular the choice of a and b), which only aims for $(\rho, \mu(\rho))$-competitiveness with $\mu(\rho) = \sup_\tau \mu_\tau(\rho)$.

Combining Theorem 1, Theorem 3 and Lemma 7, we get:

Corollary 8. *For every $\rho \in [1, \frac{e}{e-1}]$, there is a $(\rho, \mu(\rho))$-competitive algorithm for DPM.*

To prove Theorem 3, it suffices to describe a (ρ, μ)-competitive algorithm for the special case of DPM with a single idle period: Running such an algorithm for each individual period yields a (ρ, μ)-competitive algorithm for DPM with any number of idle periods, since we can simply sum inequality (1) over all periods to obtain the corresponding inequality for the entire instance.

Consider now a single idle period of length ℓ for DPM. We first recall some observations of Irani et al. [25] about the optimal *offline* algorithm: It is easy to see that the optimal offline algorithm would transition to some state j only once at the beginning of the period and remain there throughout the period, paying cost $\alpha_j \ell + \beta_j$. Thus, state j is preferred over state $j - 1$ if and only if $\alpha_{j-1}\ell + \beta_{j-1} > \alpha_j \ell + \beta_j$, or equivalently $\ell > t_j := \frac{\beta_j - \beta_{j-1}}{\alpha_{j-1} - \alpha_j}$. We may assume without loss of generality that $t_1 < \cdots < t_k$: Indeed, suppose $t_{j+1} \leq t_j$, then state j is redundant because whenever j is preferred over $j - 1$, then $j + 1$ is preferred over j. Defining $t_0 := 0$ and $t_{k+1} := +\infty$, we get a partition $[0, +\infty) = \bigcup_{j=0}^k I_j$, where $I_j = [t_j, t_{j+1})$. We can then express the cost of the offline optimum as

$$\textsc{Opt} = \alpha_{j^*}\ell + \beta_{j^*}, \text{ with } j^* \text{ such that } \ell \in I_{j^*}. \tag{6}$$

In the online setting, we of course do not know ℓ. The central idea of our algorithm is to simulate k ski rental algorithms $\mathcal{A}_1, \ldots, \mathcal{A}_k$ in parallel, where the task of $\mathcal{A}_j$ is to decide whether it is time to transition from the state $j-1$ to j. For this, we choose $\mathcal{A}_j$ to be an algorithm for ski rental with rental cost $\alpha_{j-1} - \alpha_j$ and buying cost $\beta_j - \beta_{j-1}$. Let F_τ be the CDF of the buying time of a *monotone* ski rental algorithm (for $\alpha = \beta = 1$) when given prediction τ. Recalling our reduction from arbitrary α and β to the case $\alpha = \beta = 1$ in Lemma 6, the CDF of $\mathcal{A}_j$ is given by

$$F^j(t) := F_{\tau/t_j}\left(t/t_j\right). \tag{7}$$

An outline of our algorithm is given in Algorithm 1.

Algorithm 1: DPM with a single idle period

for j=*1,...,k* **do**
  Let F^j be as defined by (7), induced by a monotone (ρ, μ)-competitive ski rental algorithm;
Choose $p \in [0, 1]$ uniformly at random;
At any time t: choose state $j = \max\{j : F^j(t) \geq p\}$;

The proof that Algorithm 1 is (ρ, μ)-competitive relies on the fact that $F^j(t)$ is non-increasing in j:

$$F^{j-1}(t) = F_{\tau/t_{j-1}}\left(t/t_{j-1}\right) \geq F_{\tau/t_j}\left(t/t_{j-1}\right) \geq F_{\tau/t_j}\left(t/t_j\right) = F^j(t),$$

where we used $t_{j-1} < t_j$ in both inequalities, the first inequality uses monotonicity of the ski rental algorithm and the second inequality uses that any CDF is non-decreasing. Thus, algorithm $\mathcal{A}_j$ signals transitioning from state $j-1$ to j no earlier than $\mathcal{A}_{j-1}$ signals transitioning from state $j-2$ to $j-1$.

A formal proof of Theorem 3 is given in the supplementary material.

4 Finding the best trade-off online

Our goal is to design an algorithm whose performance almost matches that of Corollary 8 simultaneously for *all* ρ, proving Theorem 5. It will be useful to view DPM as a Metrical Task System.

Metrical Task Systems (MTS). Metrical Task Systems (MTS), introduced by Borodin et al. [13], is a broad class of online problems containing many other problems as special cases. In MTS, we are given a metric space M of *states*. We start at a predefined initial state x_0. At each time $t = 1, 2, \ldots, T$, we are presented with a *cost function* $c_t : M \to \mathbb{R}_+$. Then, we have to choose our new state x_t and pay $\mathrm{dist}(x_{t-1}, x_t) + c_t(x_t)$, where $\mathrm{dist}(x_{t-1}, x_t)$ is the distance between x_{t-1} and x_t in M. The objective is to minimize the overall cost incurred over time.

To formulate DPM as a Metrical Task System, we choose states $0, 1, \ldots, k$ corresponding to the power states, with distances $\mathrm{dist}(i, j) = \frac{1}{2}|\beta_i - \beta_j|$, so that the cost of switching from the state 0 to j and back is β_j. We choose 0 as the initial state. We discretize time in the DPM instance using time steps of some small length $\delta > 0$. At each time step belonging to some idle period, we issue a cost function c such that $c(j) = \delta\alpha_j$ for each $j = 0, \ldots, k$. At the end of each idle period, we issue a cost function where $c(0) = 0$ and $c(j) = +\infty$ for $j = 1, \ldots, k$, which forces any algorithm to move back to the active state.

We use the result of Blum and Burch [12] to combine multiple instances of our algorithm with different parameters ρ.

Theorem 9 (Blum and Burch [12])**.** *There is an algorithm which, given N online algorithms $A_1, \ldots A_N$ for an MTS with diameter D and $\epsilon_1 < 1/2$, achieves expected cost at most*

$$(1 + \epsilon_1) \cdot \min_i\{cost(A_i)\} + O(D/\epsilon_1) \ln N.$$

Using this result, the straightforward proof of Theorem 5 is given in the supplementary material. Here, we just note that we choose a suitable set $P \subset [1, \frac{e}{e-1}]$ of size $O(1/\epsilon_2)$ so that the combination of our $(\rho, \mu(\rho))$-competitive algorithms for all $\rho \in P$ using the algorithm of Blum and Burch [12] achieves expected cost at most

$$(1 + \epsilon_1)(1 + \epsilon_2) \min_{\rho \in [1, e/(e-1)]} \left\{\rho\, \mathrm{OPT} + \mu(\rho)\eta\right\} + O\left(\frac{\beta_k}{\epsilon_1} \cdot \ln \frac{1}{\epsilon_2}\right).$$

In the supplementary material, we also argue how using results on *shifting/dynamic* regret [15; 14; 20; 21] can be used to achieve cost comparable not only to the algorithm with the best fixed ρ, but also to the best strategy of switching between multiple values of ρ a bounded number of times. This can be useful in scenarios where well-predictable parts of the input are interleaved with unpredictable or adversarial sequences.

5 Experiments

We illustrate the performance achieved by our algorithms compared to the existing learning-augmented ski-rental algorithms on the ski-rental and DPM problem on a synthetic dataset (applying previous ski rental algorithms to DPM as discussed in Sections 3 and 4). These results suggest that the performance of learning-augmented algorithms indeed degrades slowly when the error increases, providing solutions which are better than naive algorithms trusting the predictions and online algorithm for medium errors. More settings are considered in the supplementary material.

In this section we expose the results obtained on the dataset used in Purohit et al. [35]: requests are drawn uniformly at random in $[0, 4]$ and the prediction is equal to the exact request plus a random noise drawn from a normal distribution of mean 0 and standard deviation σ (rounding any negative predictions to 0). The performance is shown as the competitive ratio observed in function of σ.

For the ski-rental problem, in addition to the classical $e/(e-1)$-competitive online algorithm, we consider the algorithms FTP that blindly follows the prediction (i.e., either buy at time 0 or never), PSK, the randomized algorithm from [35], and ADJKR, the deterministic algorithm from [4]. As three algorithms depend on a hyperparameter, we determine it in order to obtain the same consistency (ρ in the notation of this paper), which is then indicated in the legend of Figure 3 (left). A consistency of 1.216 corresponds to a parameter $\lambda \approx \ln(3/2)$ for PSK, as selected in [35]. We considered 100,000 independent trials to build the dataset and ran each algorithm 10 times on it. As the maximum standard deviation is smaller than 0.01, we do not print error bars on the charts and only display the average result.

In Figure 3 (left), the results are plotted for several values of the consistency ρ for the ski-rental problem. We can observe that, on this dataset, our algorithm obtains the best performance. The relative performance between algorithms highly depends on the dataset considered. For instance, PSK will respect its consistency bound as long as FTP would be optimal, but its performance will degrade faster than our algorithm once the prediction leads to a suboptimal FTP.

The performance of the main algorithm of this paper is illustrated in Figure 3 (right). This figure uses the same dataset, interpreted as a single long instance for DPM. We consider four power-management states described by respective power consumption of $\{1, 0.47, 0.105, 0\}$ and wake-up costs of $\{0, 0.12, 0.33, 1\}$, which correspond to the active, idle, stand-by and sleep states presented in Irani et al. [25]. We use Theorem 3 to convert ski-rental algorithms to this setting. We ran each learning-augmented algorithm under the framework exposed in Section 4, using $\varepsilon_1 = 0.1$ and combining algorithms with $\rho \in \{1, 1.16, \frac{e}{e-1}\}$: values 1 and $\frac{e}{e-1}$ correspond to FTP and the standard online algorithm; $\rho = 1.16$ is either our ski rental algorithm or PSK with corresponding consistency. (FTP is only combined with the standard online algorithm.)

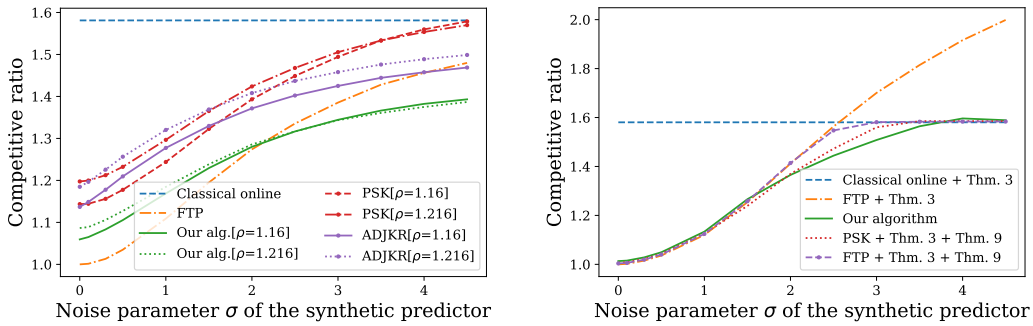

Figure 3: Performance achieved for ski-rental algorithms (left) and multiple-window multiple power management state algorithms (right).

References

[1] S. Albers. Energy-efficient algorithms. *Commun. ACM*, 53(5):86–96, 2010.

[2] S. Albers. On energy conservation in data centers. *ACM Trans. Parallel Comput.*, 6(3): 13:1–13:26, 2019.

[3] S. Albers and J. Quedenfeld. Optimal algorithms for right-sizing data centers. In C. Scheideler and J. T. Fineman, editors, *Proceedings of SPAA'18,*, pages 363–372. ACM, 2018.

[4] S. Angelopoulos, C. Dürr, S. Jin, S. Kamali, and M. Renault. Online Computation with Untrusted Advice. In *Proceedings of ITCS'20*, volume 151, pages 52:1–52:15, 2020.

[5] A. Antoniadis, C. Coester, M. Eliás, A. Polak, and B. Simon. Online metric algorithms with untrusted predictions. In *Proceedings of ICML'20*, pages 345–355. PMLR, 2020.

[6] A. Antoniadis, N. Garg, G. Kumar, and N. Kumar. Parallel machine scheduling to minimize energy consumption. In *Proceedings of SODA'20*, pages 2758–2769. SIAM, 2020.

[7] A. Antoniadis, T. Gouleakis, P. Kleer, and P. Kolev. Secretary and online matching problems with machine learned advice. In *Proceedings of NeurIPS'20*, 2020.

[8] J. Augustine, S. Irani, and C. Swamy. Optimal power-down strategies. *SIAM J. Comput.*, 37(5): 1499–1516, 2008.

[9] É. Bamas, A. Maggiori, L. Rohwedder, and O. Svensson. Learning augmented energy minimization via speed scaling. In *Proceedings of NeurIPS'20*, 2020.

[10] P. Baptiste, M. Chrobak, and C. Dürr. Polynomial-time algorithms for minimum energy scheduling. *ACM Trans. Algorithms*, 8(3):26:1–26:29, 2012.

[11] L. Benini, A. Bogliolo, and G. D. Micheli. A survey of design techniques for system-level dynamic power management. *IEEE Trans. Very Large Scale Integr. Syst.*, 8(3):299–316, 2000.

[12] A. Blum and C. Burch. On-line learning and the metrical task system problem. *Machine Learning*, 39(1):35–58, 2000.

[13] A. Borodin, N. Linial, and M. E. Saks. An optimal on-line algorithm for metrical task system. *J. ACM*, 39(4):745–763, 1992.

[14] N. Cesa-Bianchi, P. Gaillard, G. Lugosi, and G. Stoltz. A new look at shifting regret. *CoRR*, abs/1202.3323, 2012. URL `http://arxiv.org/abs/1202.3323`.

[15] N. Chen, G. Goel, and A. Wierman. Smoothed online convex optimization in high dimensions via online balanced descent. In *Proceedings of COLT'18*, volume 75, pages 1574–1594. PMLR, 2018.

[16] E. Chung, L. Benini, A. Bogliolo, Y. Lu, and G. D. Micheli. Dynamic power management for nonstationary service requests. *IEEE Trans. Computers*, 51(11):1345–1361, 2002.

[17] J. R. J. Dirk Blevins, Michael Loewe. White Paper, Inter Corporation: Designing Systems without a Suspend Supply. `https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/systems-without-suspend-supply-paper.pdf`, 2008. [Online; accessed 14-May-2021].

[18] P. Dütting, S. Lattanzi, R. P. Leme, and S. Vassilvitskii. Secretaries with advice. *CoRR*, abs/2011.06726, 2020.

[19] S. Gollapudi and D. Panigrahi. Online algorithms for rent-or-buy with expert advice. In *Proceedings of ICML'19*, pages 2319–2327, 2019.

[20] E. Hall and R. Willett. Dynamical models and tracking regret in online convex programming. In *Proceedings of ICML'13*, volume 28:1, pages 579–587. PMLR, 2013.

[21] M. Herbster and M. K. Warmuth. Tracking the best expert. In *Proceedings of ICML'95*, pages 286–294. Morgan Kaufmann, 1995.

[22] Hewlett-Packard Corporation, Intel Corporation, Microsoft Corporation, Phoenix Technologies Ltd., and Toshiba Corporation. Advanced Configuration and Power Interface Specification (ACPI). https://uefi.org/acpi/specs, 2013. [Online; accessed 14-May-2021].

[23] S. Irani and A. R. Karlin. *Online Computation*, page 521–564. PWS Publishing Co., USA, 1996. ISBN 0534949681.

[24] S. Irani and K. Pruhs. Algorithmic problems in power management. *SIGACT News*, 36(2): 63–76, 2005.

[25] S. Irani, S. Shukla, and R. Gupta. Online strategies for dynamic power management in systems with multiple power-saving states. *ACM Trans. Embed. Comput. Syst.*, 2(3):325–346, 2003.

[26] S. Irani, S. K. Shukla, and R. Gupta. Algorithms for power savings. *ACM Trans. Algorithms*, 3 (4):41, 2007.

[27] A. R. Karlin, M. S. Manasse, L. A. McGeoch, and S. S. Owicki. Competitive randomized algorithms for nonuniform problems. *Algorithmica*, 11(6):542–571, 1994.

[28] S. Khuller, J. Li, and B. Saha. Energy efficient scheduling via partial shutdown. In *SODA*, pages 1360–1372. SIAM, 2010.

[29] S. Lattanzi, T. Lavastida, B. Moseley, and S. Vassilvitskii. Online scheduling via learned weights. In *Proceedings of SODA'20*, pages 1859–1877, 2020.

[30] J. Li and S. Khuller. Generalized machine activation problems. In *Proceedings of SODA'11*, pages 80–94. SIAM, 2011.

[31] A. Lindermayr, N. Megow, and B. Simon. Double coverage with machine-learned advice. *CoRR*, abs/2103.01640, 2021. URL https://arxiv.org/abs/2103.01640.

[32] T. Lykouris and S. Vassilvitskii. Competitive caching with machine learned advice. In *Proceedings of ICML'18*, pages 3302–3311, 2018.

[33] M. Mitzenmacher and S. Vassilvitskii. Algorithms with predictions. In *Beyond the Worst-Case Analysis of Algorithms*, pages 646–662. Cambridge University Press, 2020.

[34] S. Phillips and J. Westbrook. *On-line algorithms: Competitive analysis and beyond.*, chapter 10. CRC Press, 1999.

[35] M. Purohit, Z. Svitkina, and R. Kumar. Improving online algorithms via ML predictions. In *Proceedings of NeurIPS'18*, pages 9684–9693, 2018.

[36] M. Rareshide. Power in the data center and its cost across the U.S. https://info.siteselectiongroup.com/blog/power-in-the-data-center-and-its-costs-across-the-united-states, 2017. [Online, acessed 23-May-2021].

[37] D. Rohatgi. Near-optimal bounds for online caching with machine learned advice. In *Proceedings of SODA'20*, pages 1834–1845, 2020.

[38] A. Wei. Better and simpler learning-augmented online caching. In *Proceedings of APPROX/RANDOM'20*, volume 176 of *LIPIcs*, pages 60:1–60:17, 2020.

[39] A. Wei and F. Zhang. Optimal robustness-consistency trade-offs for learning-augmented online algorithms. In *Proceedings of NeurIPS'20*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] The formal setup defined in our paper describes the limitations of our results.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] There is a (very brief) discussion just before section 1.1.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] They are mostly contained in the supplementary material.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We include it in the supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A] Our work does not include training of any models.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] As mentioned in the paper, since the maximum standard deviation is smaller than 0.01, we do not print error bars on the charts and only display the average result.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A] Our simulations can be run on a personal computer.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A] Our datasets are randomly generated.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We include the code necessary to generate the plots included in the paper.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]