The Online Pause and Resume Problem: *Optimal Algorithms and An Application to Carbon-Aware Load Shifting*

ADAM LECHOWICZ, University of Massachusetts Amherst, USA
NICOLAS CHRISTIANSON, California Institute of Technology, USA
JINHANG ZUO, California Institute of Technology & University of Massachusetts Amherst, USA
NOMAN BASHIR, University of Massachusetts Amherst, USA
MOHAMMAD HAJIESMAILI, University of Massachusetts Amherst, USA
ADAM WIERMAN, California Institute of Technology, USA
PRASHANT SHENOY, University of Massachusetts Amherst, USA

We introduce and study the online pause and resume problem. In this problem, a player attempts to find the k lowest (alternatively, highest) prices in a sequence of fixed length T, which is revealed sequentially. At each time step, the player is presented with a price and decides whether to accept or reject it. The player incurs a *switching cost* whenever their decision changes in consecutive time steps, i.e., whenever they pause or resume purchasing. This online problem is motivated by the goal of carbon-aware load shifting, where a workload may be paused during periods of high carbon intensity and resumed during periods of low carbon intensity and incurs a cost when saving or restoring its state. It has strong connections to existing problems studied in the literature on online optimization, though it introduces unique technical challenges that prevent the direct application of existing algorithms. Extending prior work on threshold-based algorithms, we introduce *double-threshold* algorithms for both the minimization and maximization variants of this problem. We further show that the competitive ratios achieved by these algorithms are the best achievable by any deterministic online algorithm. Finally, we empirically validate our proposed algorithm through case studies on the application of carbon-aware load shifting using real carbon trace data and existing baseline algorithms.

CCS Concepts: • **Theory of computation** → *Online algorithms*; *Theory and algorithms for application domains*.

Additional Key Words and Phrases: online pause and resume, carbon-aware load shifting, online algorithms, switching costs, k-search

45

Authors' addresses: Adam Lechowicz, University of Massachusetts Amherst, USA, alechowicz@cs.umass.edu; Nicolas Christianson, California Institute of Technology, USA, nchristianson@caltech.edu; Jinhang Zuo, California Institute of Technology & University of Massachusetts Amherst, USA, jhzuo@cs.umass.edu; Noman Bashir, University of Massachusetts Amherst, USA, nbashir@cs.umass.edu; Mohammad Hajiesmaili, University of Massachusetts Amherst, USA, hajiesmaili@cs.umass.edu; Adam Wierman, California Institute of Technology, USA, adamw@caltech.edu; Prashant Shenoy, University of Massachusetts Amherst, USA, shenoy@cs.umass.edu.

1 INTRODUCTION

This paper introduces and studies the *online pause and resume problem* (OPR), considering both minimization (OPR-min) and maximization (OPR-max) variants. In OPR-min, a player is presented with time-varying prices in a sequential manner and decides whether or not to purchase one unit of an item at the current price. The player must purchase k units of the item over a time horizon of T and they incur a *switching cost* whenever their decision changes in consecutive time steps, i.e., whenever they pause or resume purchasing. The goal of the player is to minimize their total cost, which consists of the aggregate price of purchasing k units and the aggregate switching cost incurred over T slots. In OPR-max, the setting is exactly the same, but the goal of the player is to maximize their total profit, and any switching cost that they incur is subtracted. In both cases, the price values are revealed to the player one by one in an online manner, and the player has to make a decision without knowing the future values.

Our primary motivation for introducing OPR is the emerging importance of carbon-aware computing and, more specifically, carbon-aware temporal workload shifting, which has seen significant attention in recent years [1, 6, 36, 47]. In carbon-aware temporal workload shifting, an interruptible and deferrable workload may be paused during periods of high carbon intensity and resumed during periods of low carbon intensity. The workload must be running for k units of time to complete and must be completed before its deadline T. However, pausing and resuming the workload typically comes with overheads such as storing the state in memory and checkpointing. For example, an empirical study [20] shows that this overhead can nullify any savings in carbon emissions from temporal shifting if the job is interrupted frequently. Moreover, with the rise of big ML training workloads, such as the training, fine-tuning, and inference of large language models (LLM), data center workloads' memory footprints are frequently in the hundreds of GBs [37, 41, 48]. These emerging workloads will result in high checkpoint-and-restore overheads, which must be considered in carbon-aware scheduling. This motivates adding a *switching cost* in OPR, since a naïve algorithm that does not account for the interruptions' overhead may frequently checkpoint and, in some cases, increase carbon emissions beyond a carbon-agnostic execution.

The objective of temporal workload shifting is to minimize the total carbon footprint of running the workload, which includes both the original compute demand and the overhead due to pausing and resuming (a.k.a., the switching cost). We consider a worst-case performance objective based on competitive analysis, defined explicitly in Section 2, wherein we seek to find an effective algorithm that is robust to uncertain and nonstochastic fluctuations in price (or carbon intensity in the context of carbon-aware load shifting). We note that even though statistical modeling of grid carbon intensity has been explored [31], we focus on developing worst-case optimized algorithms as the intended application has nonlinearity and nonstationarity, which complicate the task of designing a single probabilistic model to solve this problem.

OPR also captures other interesting applications with highly variable time-varying costs where switching frequently is undesirable. A related example is the carbon-aware electric vehicle (EV) charging problem, which considers when to charge an EV with respect to the time-varying availability of carbon-free electricity, a charging deadline (e.g., set by the EV owner), and battery health design goals (i.e., a constant charging rate is better for battery longevity) [52]. When the charger is *non-adaptive* (i.e., the charging rate is either 0 or the maximum rate), the problem reduces exactly to OPR. Beyond these "carbon-aware" applications, there are additional examples that deal with pricing, such as managing grid-scale energy storage with respect to real-time prices in the wholesale electricity market, where a "smooth" charge or discharge rate is desired [49]. Another example is renting spot virtual machines from a cloud service provider in the setting where pricing is set according to supply-demand dynamics [2, 40, 53].

On the theory front, the OPR problem has strong connections to various existing problems in the literature on online optimization. We extensively review the prior literature in Section 7 and focus on the most relevant theoretical problems below. The OPR problem is a generalization of the k-search problem [23, 29], which belongs to the broader class of online conversion problems [44], a.k.a., time series search and one-way trading [15]. In the minimization variant of the k-search problem, an online decision-maker aims to buy k units of an item for the least cost over a sequence of time-varying cost values. At each step, a cost value is observed, and the decision is whether or not to buy one unit at the current observed cost without knowing the future values (see Section 2.2 for a deeper discusssion of k-search). In contrast to k-search, the OPR problem introduces the additional component of managing the switching cost, which poses a significant additional challenge in algorithm design.

The existence of the switching cost in OPR connects it to the well-studied problem of smoothed online convex optimization (SOCO) [25], also known as convex function chasing (CFC) [17], and its generalizations including metrical task systems (MTS) [7]. In SOCO, a learner is faced with a sequence of cost functions f_t that are revealed online, and must choose an action x_t after observing f_t. Based on that decision, the learner incurs a hitting cost, $f_t(x_t)$ as well as a switching cost, $\|x_t - x_{t-1}\|$, which captures the cost associated with changing the decision between rounds. In contrast to SOCO, OPR includes the long-term constraint of satisfying the demand of k units over the horizon T, which poses a significant challenge not present in SOCO-like problems.

The coexistence of these differentiating factors, namely the *switching cost* and the *long-term deadline constraint*, make OPR uniquely challenging, and means that prior algorithms and analyses for related problems such as k-search and SOCO cannot be directly adapted.

Contributions. We introduce online algorithms for the minimization and maximization variants of OPR and show that our algorithms achieve the best possible competitive ratios. We also evaluate the empirical performance of the proposed algorithms on a case study of carbon-aware load shifting. The details of our contributions are outlined below.

Algorithmic idea: Double-threshold. To tackle OPR, we focus our efforts on online threshold-based algorithms (OTA), the prominent design paradigm for classic problems such as k-search [23, 29], one-way trading [15, 44], and online knapsack problems [45, 50, 54]. In the k-min search problem, for example, a threshold-based algorithm specifies k threshold values and chooses to trade the i-th item only if the current price is less than or equal to the value suggested by the i-th threshold value.

Direct application of prior OTA algorithms to OPR results in undesirable behavior (such as frequently changing decisions) since their threshold function design is oblivious to the switching cost present in OPR. To address this challenge, we seek an algorithm that can simultaneously achieve the following behaviors: (1) when the player is in "trading mode," they should not impulsively switch away from trading in response to a price that is only slightly worse, since this will result in a switching penalty; and (2) the player should not switch to "trading mode" unless prices are sufficiently good to warrant the switching cost. These two ideas motivate an algorithm design that uses two distinct threshold functions, each of which captures one of the above two cases. We present our algorithms DTPR-min and DTPR-max for OPR-min and OPR-max, respectively, in Section 3, which build upon this high-level idea of a double-threshold.

Main results. While OTA algorithms are intuitive and simple to describe, it is highly challenging to design threshold functions that lead the corresponding algorithms to be competitive against the offline optimum. The addition of switching cost in OPR further exacerbates the technical challenge of designing optimal threshold functions. The key result which enables our double-threshold approach is a technical observation (see Observation 3), which shows that the difference between

the functions guiding the algorithm's decisions should be exactly 2β, where β represents the fixed switching cost incurred by changing the decision in OPR.

Identifying this relationship between the two threshold functions significantly facilitates the competitive analysis of both DTPR-min and DTPR-max, enabling our derivation of a closed form of each threshold. Using this idea, we characterize the competitive ratios of DTPR-min and DTPR-max as a function of problem parameters, including an explicit dependence on the magnitude of the switching cost β (see Theorems 4 and 5). Furthermore, we derive lower bounds for the competitive ratio of any deterministic online algorithm, showing that our proposed algorithms are optimal for this problem (formal statements in Theorems 8 and 9). The competitive ratios we derive for both DTPR-min and DTPR-max exactly recover the best prior competitive results for the k-search problem [29], which corresponds to the case of $\beta = 0$ in OPR, i.e., no switching cost. Formal statements and a more detailed discussion of our main results are presented in Section 4.

Case study. Finally, in Section 6, we illustrate the performance of our proposed algorithm by conducting an experimental case study simulating the carbon-aware load shifting problem. We utilize real-world *carbon traces* from Electricity Maps [32], which contain carbon intensity values for grid-sourced electricity across the world. Our experiments simulate different strategies for scheduling a deferrable and interruptible workload in the face of uncertain future carbon intensity values. We show that our algorithm's performance significantly improves upon existing baseline methods and adapted forms of algorithms for related problems such as k-min search.

2 PROBLEM FORMULATION AND PRELIMINARIES

We begin by formally introducing the OPR problem and providing background on the online threshold-based algorithm design paradigm, which is used in the design of our proposed algorithms. Table 1 summarizes the core notations for OPR. Recall that this formulation is motivated by the setting of carbon-aware temporal workload shifting, as described in the introduction.

2.1 Problem Formulation

We present two variants of the online pause and resume problem (OPR).[1] In OPR-min (OPR-max) a player must buy (sell) $k \geq 1$ units of some asset (one unit at each time step) with the goal of minimizing (maximizing) their total cost (profit) within a time horizon of length T. At each time step $1 \leq t \leq T$, the player is presented with a price c_t, and must immediately decide whether to accept this price ($x_t = 1$) or reject it ($x_t = 0$). The player is required to complete this transaction for all k units by some point in time T. Both k and T are known in advance. Thus, the requirement of k transactions is a hard constraint, i.e., $\sum_{t=1}^{T} x_t = k$, and if at time $T - i$ the player still has i units remaining to buy/sell, they must accept the prices in the subsequent i slots to accomplish k transactions.

Additionally, in both variants of OPR, the player incurs a *fixed switching cost* $\beta > 0$ whenever they decide to change decisions between two adjacent time steps (i.e., when $\|x_{t-1} - x_t\| = 1$). We assume that $x_0 = 0$ and $x_{T+1} = 0$, implying that any player must incur a minimum switching cost of 2β, once for switching "on" and once for switching "off". While the player incurs at least a switching cost of 2β, note that the total switching cost incurred by the player is bounded by the size of the asset k since the switching cost cannot be larger than $k2\beta$.

---

[1]We use OPR whenever the context is applicable to both minimization (OPR-min) and maximization (OPR-max) variants of the problem, otherwise, we refer to the specific variant. The same policy applies to DTPR, our proposed algorithm for OPR.

Table 1. A summary of key notations

Notation	Description
$k \in \mathbb{N}$	Number of units which must be bought (or sold)
T	Deadline constraint; the player must buy (or sell) k units before time T
$t \in [1, T]$	Current time step
$x_t \in \{0, 1\}$	Decision at time t. $x_t = 1$ if price c_t is accepted, $x_t = 0$ if c_t is not accepted
β	Switching cost incurred when algorithm's decision $x_t \neq x_{t-1}$
U	Upper bound on any price that will be encountered
L	Lower bound on any price that will be encountered
$\theta = U/L$	Price fluctuation ratio
c_t	(*Online input*) Price revealed to the player at time t
$c_{\min}$ & $c_{\max}$	(*Online input*) The actual minimum and maximum prices in a sequence

In summary, the offline version of OPR-min can be summarized as follows:

$$\min \underbrace{\left(\sum_{t=1}^{T} c_t x_t\right)}_{\text{Accepted prices}} + \underbrace{\left(\sum_{t=1}^{T+1} \beta ||x_t - x_{t-1}||\right)}_{\text{Switching cost}}, \quad \text{s.t.,} \quad \underbrace{\sum_{t=1}^{T} x_t = k}_{\text{Deadline constraint}}, \quad x_t \in \{0, 1\}, \ \forall t \in [1, T], \quad (1)$$

while the offline version of OPR-max is

$$\max \left(\sum_{t=1}^{T} c_t x_t\right) - \left(\sum_{t=1}^{T+1} \beta ||x_t - x_{t-1}||\right), \quad \text{s.t.,} \quad \sum_{t=1}^{T} x_t = k, \quad x_t \in \{0, 1\}, \ \forall t \in [1, T]. \quad (2)$$

Of course, our focus is the online version of OPR, where the player must make irrevocable decisions at each time step without the knowledge of future inputs. More specifically, in both variants of OPR the sequence of prices $\{c_t\}_{t \in [1,T]}$ is revealed sequentially – future prices are *unknown* to an online algorithm, and each decision x_t is irrevocable.

Competitive analysis. Our goal is to design an online algorithm that maintains a small *competitive ratio* [7], i.e., performs nearly as well as the offline optimal solution. For an online algorithm ALG and an offline optimal solution OPT, the competitive ratio for a minimization problem is defined as: $\text{CR}(\text{ALG}) = \max_{\sigma \in \Omega} \text{ALG}(\sigma)/\text{OPT}(\sigma)$, where σ denotes a valid input sequence for the problem and Ω is the set of all feasible input instances. Further, $\text{OPT}(\sigma)$ is the optimal cost given this input, and $\text{ALG}(\sigma)$ is the cost of the solution obtained by running the online algorithm over this input. Conversely, for a problem with a maximization objective, the competitive ratio is defined as $\max_{\sigma \in \Omega} \text{OPT}(\sigma)/\text{ALG}(\sigma)$. With these definitions, the competitive ratio for both minimization and maximization problems is always greater than or equal to one, and the lower the better.

Note that competitive algorithm development, in its classic worst-case optimized design, cannot capture data-driven adaptation and stochasticity of data in decision-making. However, beyond the significance of the theoretical analysis in this framework, competitive algorithms could be of interest to practitioners since they are robust against adversarial or non-stationary behavior in the underlying environment. For example, in the context of carbon-aware load shifting, the carbon intensity values significantly change across the temporal and spatial domains following the makeup and behavior of an electric grid (e.g., different ISOs and generation mixes; see Figure 9 in the appendix); and online algorithms are robust to those drastic temporal and spatial variations. Competitive algorithms are extremely simple to implement, e.g., in OPR, all we need are two threshold functions to decide the pause and resume decisions. Furthermore, worst-case optimized

algorithms can potentially be augmented with machine-learned predictions, as explored in e.g. [3, 4, 12, 30, 35, 44], to achieve the best of both worlds of worst-case and average-case performance.

Assumptions and additional notations. We make no assumptions on the underlying distribution of the prices other than the assumption that the set of prices arriving online $\{c_t\}_{t\in[1,T]}$ has bounded support, i.e., $c_t \in [L, U]$ $\forall t \in [1, T]$, where L and U are known to the player. We also define $\theta = U/L$ as the *price fluctuation*. These are standard assumptions in the literature for many online problems, including one-way trading, online search, and online knapsack; and without them the competitive ratio of any algorithm is unbounded. Most papers in this literature additionally assume that $U, L > 0$ (i.e. the lowest price is still positive), but our design can handle the special case where $L = 0$, and therefore do not adopt this assumption. We use $c_{\min}(\sigma) = \min_{t\in[1,T]} c_t$ and $c_{\max}(\sigma) = \max_{t\in[1,T]} c_t$ to denote the minimum and maximum encountered prices for any valid OPR sequence σ.

2.2 Background: Online Threshold-Based Algorithms (OTA)

Online threshold-based algorithms (OTA) are a family of algorithms for online optimization in which a carefully designed *threshold function* is used to specify the decisions made at each time step. At a high level, the threshold function defines the "minimum acceptable quality" that an arriving input/price must satisfy in order to be accepted by the algorithm. The threshold is chosen specifically so that an agent greedily accepting prices meeting the threshold at each step will be ensured a competitive guarantee. This algorithmic framework has seen success in the online search and one-way trading problems [15, 23, 29, 44] as well as the related online knapsack problem [45, 50, 54]. In these works, the derived threshold functions are optimal in the sense that the competitive ratios of the resulting threshold-based algorithms match information-theoretic lower bounds of the corresponding online problems. As discussed in the introduction, the framework does not apply directly to the OPR setting, but we make use of ideas and techniques from this literature. We briefly detail the most relevant highlights from the prior results before discussing how these related problems generalize to OPR in the next section.

1-min/1-max search. In the online 1-min/1-max search problem, a player attempts to find the single lowest (respectively, highest) price in a sequence, which is revealed sequentially. The player's objective is to either minimize their cost or maximize their profit. When each price arrives, the player must decide immediately whether to accept the price, and the player is forced to accept exactly one price before the end of the sequence. For this problem, El-Yaniv et al. [15] presents a deterministic threshold-based algorithm. The algorithm assumes a finite price interval, i.e., the price is bounded by the interval $[L, U]$, where L and U are known. Then, it sets a constant threshold $\Phi = \sqrt{LU}$, and the algorithm simply selects the first price that is less than or equal to Φ (for the maximization version, it accepts the first price greater than or equal to Φ). This algorithm achieves a competitive ratio of $\sqrt{U/L} = \sqrt{\theta}$, which matches the lower bound; hence, it is optimal [15].

k-min/k-max search. The online k-min/k-max search problem extends the 1-min/1-max search problem – a player attempts to find the k lowest (conversely, highest) prices in a sequence of prices revealed sequentially. The player's objective is identical to the 1-min/1-max problem, and the player must accept at least k prices by the end of the sequence. Several works have developed a known optimal deterministic threshold-based algorithm for this problem, including [15, 29]. Leveraging the same assumption of a finite price interval $[L, U]$, the threshold function is a sequence of k thresholds $\{\Phi_i\}_{i\in[1,k]}$, which is also called the *reservation price policy*. At each step, the algorithm accepts the first price, which is less than or equal to Φ_i, where $i - 1$ is the number of prices that

have been accepted thus far (for the maximization version, it accepts the first price which is $\geq \Phi_i$). In the k-min setting, this algorithm is α-competitive, where α is the unique solution of

$$\frac{1 - 1/\theta}{1 - 1/\alpha} = \left(1 + \frac{1}{\alpha k}\right)^k. \tag{3}$$

For the k-max variant, this algorithm is ω-competitive, where ω is the unique solution of

$$\frac{\theta - 1}{\omega - 1} = \left(1 + \frac{\omega}{k}\right)^k. \tag{4}$$

The sequence of thresholds $\{\Phi_i\}_{i \in [1,k]}$ for both variants of the problem are constructed by analyzing possible input cases, "hedging" against the risk that future (unknown) prices will jump to the worst possible value, i.e., U for k-min search, L for k-max search. These potential cases can be enumerated for different values of i, where $0 \leq i \leq k$ denotes the number of prices accepted so far. By simultaneously *balancing* the competitive ratios for each of these cases (setting each ratio equal to the others), the optimal threshold values and the optimal competitive ratios are derived. We refer to this technique as the *balancing rule* and a rigorous proof of this approach, with corresponding lower bounds, can be found in [29]. The lower bounds highlight that the α and ω which solve the expressions for the competitive ratios above are optimal for any deterministic k-min and k-max search algorithms, respectively. Further, α and ω provide insight into a fundamental difference between the minimization and maximization settings of k-search. As discussed in [29], for large θ, the best algorithm for k-max search is roughly $O(k\sqrt[k]{\theta})$-competitive, while the best algorithm for k-min search is at best $O(\sqrt{\theta})$-competitive. Similarly, for fixed θ and large k, the optimal competitive ratio for k-max search is roughly $O(\ln \theta)$, while the optimal competitive ratio for k-min search converges to $O(\sqrt{\theta})$.

3 DOUBLE THRESHOLD PAUSE AND RESUME (DTPR) ALGORITHM

A fundamental challenge in algorithm design for OPR is how to characterize threshold functions that incorporate the presence of switching costs in their design. Our key algorithmic insight is to incorporate the switching cost into the threshold function by defining *two distinct threshold functions*, where the function to be used for price admittance changes based on the current state (i.e., whether or not the previous price was accepted by the algorithm).

To provide intuition for the state-dependence of the threshold function, consider the setting of OPR-min. At a high level, if the player has not accepted the previous price, they should wait to accept anything until prices are sufficiently low to justify incurring a cost to switch decisions. On the other hand, if the player has accepted the previous price, they might be willing to accept a slightly higher price – if they do not accept this price, they will incur a cost to switch decisions. While this high-level idea is intuitive, characterizing the form of threshold functions such that the resulting algorithms are competitive is challenging.

The DTPR-min *algorithm.* Our proposed algorithm, Double Threshold Pause and Resume (DTPR) for OPR-min is summarized in Algorithm 1. Prior to any prices arriving online, DTPR-min computes two families of threshold values, $\{\ell_i\}_{i \in [1,k]}$ and $\{u_i\}_{i \in [1,k]}$, where $\ell_i \leq u_i \; \forall i \in [1,k]$, whose values are defined as follows.

DEFINITION 1 (DTPR-min THRESHOLD VALUES). *For each integer i on the interval $[1,k]$, the following expressions give the corresponding threshold values of u_i and ℓ_i for* DTPR-min.

$$u_i = U\left[1 - \left(1 - \frac{1}{\alpha}\right)\left(1 + \frac{1}{k\alpha}\right)^{i-1}\right] + 2\beta\left[\left(\frac{1}{k\alpha} - \frac{1}{k} + 1\right)\left(1 + \frac{1}{k\alpha}\right)^{i-1}\right], \qquad \ell_i = u_i - 2\beta, \tag{5}$$

Algorithm 1 Double Threshold Pause and Resume for OPR-min (DTPR-min)

Input: threshold values $\{\ell_i\}_{i \in [1,k]}$ and $\{u_i\}_{i \in [1,k]}$ defined in Eq. (5), deadline T
Output: online decisions $\{x_t\}_{t \in [1,T]}$

  1: **initialize:** $i = 1$
  2: **while** price c_t arrives **and** $i \leq k$ **do**
  3:     **if** $(k - i) \geq (T - t)$ **then**   ▷ close to the deadline T, we must accept remaining prices
  4:         price c_t is accepted, set $x_t = 1$
  5:     **else if** $x_{t-1} = 0$ **then**   ▷ If previous price was not accepted
  6:         **if** $c_t \leq \ell_i$ **then**   price c_t is accepted, set $x_t = 1$
  7:         **else** price c_t is rejected, set $x_t = 0$
  8:     **else if** $x_{t-1} = 1$ **then**   ▷ If previous price was accepted
  9:         **if** $c_t \leq u_i$ **then**   price c_t is accepted, set $x_t = 1$
 10:         **else** price c_t is rejected, set $x_t = 0$
 11:     update $i = i + x_t$

---

where α is the competitive ratio of DTPR-min *defined in Equation* (9).

The role of these thresholds is to incorporate the switching cost into the algorithm's decisions, and to alter the acceptance criteria of DTPR-min based on the current state. For OPR-min, the current state is *whether the previous item was accepted*, i.e., whether x_{t-1} is 0 or 1. As prices are sequentially revealed to the algorithm at each time t, the ith price accepted by DTPR-min will be the first price which is at most ℓ_i if $x_{t-1} = 0$, or at most u_i if $x_{t-1} = 1$. We note that L does not explicitly appear in this definition. As i approaches k, the values of these thresholds decrease, getting closer to L (See Figure 1). Note that, as indicated in Line 4, DTPR-min may be forced to accept the last prices of the sequence, which can be "worse" than the current threshold values, to satisfy the deadline constraint of OPR. Since T (the deadline) does not appear explicitly in the threshold definition, our analysis can handle the case where T is not known to the online player, and the forced acceptance is triggered by some external signal.

The DTPR-max *algorithm.* Pseudocode is summarized in the appendix, in Algorithm 2. The logical flow of DTPR-max shares a similar structure to that of DTPR-min, with a few important differences highlighted here. For OPR-max, the ith price accepted by DTPR-max will be the first price which is at least u_i if $x_{t-1} = 0$, or at least ℓ_i if $x_{t-1} = 1$. Further, the threshold functions are defined as follows.

DEFINITION 2 (DTPR-max THRESHOLD VALUES). *For each integer i on the interval $[1, k]$, the following expressions give the corresponding threshold values of ℓ_i and u_i for* DTPR-max.

$$\ell_i = L \left[1 + (\omega - 1) \left(1 + \frac{\omega}{k} \right)^{i-1} \right] - 2\beta \left[\left(\frac{\omega}{k} - \frac{1}{k} + 1 \right) \left(1 + \frac{\omega}{k} \right)^{i-1} \right], \quad u_i = \ell_i + 2\beta, \quad (6)$$

where ω is the competitive ratio of DTPR-max *defined in Equation* (10).

In Figures 1 and 2, we plot threshold values for DTPR-min and DTPR-max, respectively, using example parameters of $U = 30, L = 5, k = 10$, and $\beta = 3$. We annotate the difference of 2β between ℓ_i and u_i; recall that each of these thresholds corresponds to a *current state* for DTPR, i.e. whether the previous item was accepted. Note that the DTPR-min threshold values *decrease* as k gets larger, while the DTPR-max threshold values *increase* as k gets larger. At a high-level, each ith threshold "hedges" against a scenario where none of the future prices meet the current threshold. In this case, even if the algorithm is forced to accept the *worst possible prices* at the end of the sequence, we want competitive guarantees against an offline OPT. Such guarantees rely on the fact that in
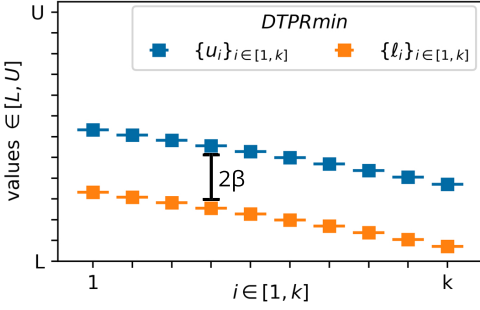
Fig. 1. DTPR-min thresholds ℓ_i and u_i for $i \in [1, k]$ plotted using example parameters ($k = 10$).
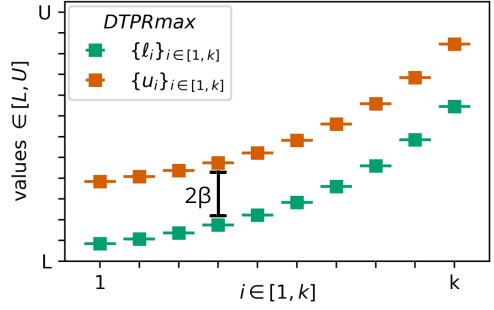
Fig. 2. DTPR-max thresholds u_i and ℓ_i for $i \in [1, k]$ plotted using example parameters ($k = 10$).

the worst-case, OPT cannot accept prices that are all significantly better than DTPR's ith "unseen" threshold value because such prices did not exist in the sequence.

Designing the Double Threshold Values

A key component of the DTPR algorithms for both variants are the thresholds in Equations (5) and (6). The key idea is to design the thresholds by incorporating the switching cost into the balancing rules as a hedge against possible worst-case scenarios. To accomplish this, we enumerate three difficult cases that DTPR may encounter. (CASE-1): Consider an input sequence where DTPR does not accept any prices before it is forced to accept the last k prices. Here, the enforced prices in the worst-case sequence will be U for OPR-min and L for OPR-max. This sequence occurs only if no price in the sequence meets the first threshold for acceptance. On the other hand, in the case that DTPR does accept prices before the end of the sequence, we can further divide the possible sequences into two extreme cases for the *switching cost* it incurs. (CASE-2): In one extreme, the algorithm incurs only the minimum switching cost of 2β, meaning that k contiguous prices are accepted by DTPR. (CASE-3): In the other extreme, DTPR incurs the maximum switching cost of $k2\beta$, meaning that k non-contiguous prices are accepted. Intuitively, in order for DTPR to be competitive in either of these extreme cases, the prices accepted in the latter case should be sufficiently "good" to absorb the extra switching cost of $(k - 1)2\beta$.

Given the insight from these cases, we use can use the balancing rule (see Section 2.2) to derive the two threshold families. Let σ be any arbitrary sequence for OPR. Given these extreme input sequences, we now concretely show how to write the balancing rule equations. We consider the cases of DTPR-min and DTPR-max separately below.

Balancing equations for DTPR-min. To balance between possible inputs for OPR-min, consider the following examples for three different values of $c_{\min}(\sigma) > \ell$, $\ell = \{\ell_1, \ell_2, \ell_3\}$. If $c_{\min}(\sigma) > \ell_i$, we know that OPT cannot do better than $k\ell_i + 2\beta$. Suppose that α is the target competitive ratio. Then each term in equation (7) corresponds to a different *case* (e.g. a possible input), and we solve for the

threshold values by "balancing" between all of these possible cases:

$$\frac{\text{DTPR-min}(\sigma)}{\text{OPT}(\sigma)} \leq \underbrace{\frac{kU + 2\beta}{k\ell_1 + 2\beta}}_{c_{\min}(\sigma) > \ell_1} = \underbrace{\frac{\ell_1 + (k-1)U + 4\beta}{k\ell_2 + 2\beta} = \frac{u_1 + (k-1)U + 2\beta}{k\ell_2 + 2\beta}}_{c_{\min}(\sigma) > \ell_2} \cdots \tag{7}$$

$$\cdots = \underbrace{\frac{\ell_1 + \ell_2 + (k-2)U + 6\beta}{k\ell_3 + 2\beta} = \frac{\ell_1 + u_2 + (k-2)U + 4\beta}{k\ell_3 + 2\beta} = \frac{u_1 + u_2 + (k-2)U + 2\beta}{k\ell_3 + 2\beta}}_{c_{\min}(\sigma) > \ell_3} = \cdots = \alpha.$$

As an example, consider $c_{\min}(\sigma) > \ell_2$ and the corresponding cases enumerated above. Suppose DTPR-min accepts one price before the end of the sequence σ, and the other prices accepted are all U. In the first case, where the competitive ratio is $\frac{\ell_1 + (k-1)U + 4\beta}{k\ell_2 + 2\beta}$, we consider the scenario where DTPR-min switches twice: once to accept the price ℓ_1, and once to accept $(k-1)$ prices at the end of the sequence, incurring switching cost of 4β.

In the second case, where the competitive ratio is $\frac{u_1 + (k-1)U + 2\beta}{k\ell_2 + 2\beta}$, we consider the hypothetical scenario where DTPR-min only switches once to accept some value u_1 followed by $(k-1)$ prices at the end of the sequence, incurring switching cost of 2β. By enumerating cases in this fashion for the other possible values of $c_{\min}(\sigma)$, we derive a relationship between the lower thresholds ℓ_i and the upper thresholds u_i in terms of the switching cost.

Balancing equations for DTPR-max. The same idea extends to balance between possible inputs for OPR-max. Consider the following examples for a few values of $c_{\max}(\sigma)$. If $c_{\max}(\sigma) < u_i$, we know that OPT cannot do better than $ku_i - 2\beta$. Suppose that ω is the target competitive ratio, and we balance between the following possible cases (e.g. possible inputs):

$$\frac{\text{OPT}(\sigma)}{\text{DTPR-max}(\sigma)} \leq \underbrace{\frac{ku_1 - 2\beta}{kL - 2\beta}}_{c_{\max}(\sigma) < u_1} = \underbrace{\frac{ku_2 - 2\beta}{u_1 + (k-1)L - 4\beta} = \frac{ku_2 - 2\beta}{\ell_1 + (k-1)L - 2\beta}}_{c_{\max}(\sigma) < u_2} \cdots \tag{8}$$

$$\cdots = \underbrace{\frac{ku_3 - 2\beta}{u_1 + u_2 + (k-2)L - 6\beta} = \frac{ku_3 - 2\beta}{u_1 + \ell_2 + (k-2)L - 4\beta} = \frac{ku_3 - 2\beta}{\ell_1 + \ell_2 + (k-2)L - 2\beta}}_{c_{\max}(\sigma) < u_3} = \cdots = \omega.$$

Solving for the threshold values. Given the above balancing equations for both the minimization and maximization variants, the next step is to solve for the unknown values of ℓ_i and u_i. The following observation summarizes the key insight that enables this. We show that one can express each ℓ_i in terms of u_i and β, which facilitates the analysis required to solve for thresholds in each balancing equation (given by Equations (7) and (8)).

OBSERVATION 3. *By letting* $u_i = \ell_i + 2\beta$ $\forall i \in [1, k]$, *we obtain each possible worst-case permutation of* ℓ_i *thresholds,* u_i *thresholds, and switching cost. Let* $y \in [1, k-1]$ *denote the number of switches incurred by* DTPR.
For DTPR-min, *suppose that* $c_{\min}(\sigma) > \ell_{j+1}$. *By the definition of* DTPR-min, *we know that accepting any* u_i *helps avoid a switching cost of* $+2\beta$ *in the worst case. Thus,*

$$\sum_{i=0}^{j} u_i + (k-j)U + 2\beta = \underbrace{\ell_i + \ldots}_{y} + \underbrace{u_i + \ldots}_{j-y} + (k-j)U + (y+1)2\beta = \sum_{i=0}^{j} \ell_i + (k-j)U + (j+1)2\beta.$$

For DTPR-max, *suppose that* $c_{\max}(\sigma) < u_{j+1}$. *By the definition of* DTPR-max, *we know that accepting any* ℓ_i *helps avoid a switching cost of* -2β *in the worst case. Thus,*

$$\sum_{i=0}^{j} \ell_i + (k-j)L - 2\beta = \underbrace{u_i + \ldots}_{y} + \underbrace{\ell_i + \ldots}_{j-y} + (k-j)L - (y+1)2\beta = \sum_{i=0}^{j} u_i + (k-j)L - (j+1)2\beta.$$

With the above observation, for DTPR-min, one can substitute $u_i - 2\beta$ for each ℓ_i. By comparing adjacent terms in Equation (7), standard algebraic manipulations give a closed form for each u_i in terms of u_1. Setting $\frac{kU+2\beta}{k(u_1-2\beta)+2\beta} = \alpha$, we obtain the explicit expression for u_1, yielding a closed formula for $\{u_i\}_{i\in[1,k]}$ and $\{\ell_i\}_{i\in[1,k]}$ in Equation (5). Considering the balancing rule in Equation (7) for the case where $c_{\min}(\sigma) \geq \ell_{k+1}$, it follows that $\ell_{k+1} = L$, and thus $u_{k+1} = L + 2\beta$. By substituting this value into Definition 1, we obtain an explicit expression for α as shown in Equation (9).

Conversely, for DTPR-max, we substitute $\ell_i + 2\beta$ for each u_i. By comparing adjacent terms in Equation (8), standard methods give a closed form for each ℓ_i in terms of ℓ_1. Setting $\frac{k(\ell_1+2\beta)-2\beta}{kL-2\beta} = \omega$, we obtain the explicit expression for ℓ_1, yielding the closed formula for $\{\ell_i\}_{i\in[1,k]}$ and $\{u_i\}_{i\in[1,k]}$ in Equation (6). Considering the balancing rule in Equation (8) for the case where $c_{\max}(\sigma) \leq u_{k+1}$, it follows that $u_{k+1} = U$, and thus $\ell_{k+1} = U - 2\beta$. By substituting this value into Definition 2, we obtain an explicit expression for ω as shown in Equation (10).

4 MAIN RESULTS

We now present competitive results of DTPR for both variants of OPR and discuss the significance of the results in relation to other algorithms for related problems. Our results for the competitive ratios of DTPR-min and DTPR-max are summarized in Theorems 4 and 5. We also state the lower bound results for any deterministic online algorithms for OPR-min and OPR-max in Theorems 8 and 9. Proofs of the results for DTPR-min and DTPR-max are deferred to Section 5 and Appendix B, respectively. Formal proofs of lower bound theorems are given in Appendix D, and a sketch is shown in Section 5.2. Note that in the competitive results, $W(x)$ denotes the Lambert W function, i.e., the inverse of $f(x) = xe^x$. It is well-known that $W(x)$ behaves like $\ln(x)$ [21, 43]. We start by presenting our competitive bounds on DTPR-min and DTPR-max.

THEOREM 4. DTPR-min *is an* α-*competitive deterministic algorithm for* OPR-min, *where* α *is the unique positive solution of*

$$\frac{U - L - 2\beta}{U(1 - 1/\alpha) - \left(2\beta - \frac{2\beta}{k} + \frac{2\beta}{k\alpha}\right)} = \left(1 + \frac{1}{k\alpha}\right)^k. \tag{9}$$

THEOREM 5. DTPR-max *is an* ω-*competitive deterministic algorithm for* OPR-max, *where* ω *is the unique positive solution of*

$$\frac{U - L - 2\beta}{L(\omega - 1) - 2\beta\left(1 - \frac{1}{k} + \frac{\omega}{k}\right)} = \left(1 + \frac{\omega}{k}\right)^k. \tag{10}$$

These theorems present upper bounds on the competitive ratios, showing their dependence on the problem parameters. To investigate the behavior of these competitive ratios, in Figures 3 and 4, we show the competitive ratios of both algorithms as problem parameters are varied. More specifically, in Figure 3, we visualize α as a function of β and L, where k and U are fixed. The color (shown as an annotated color bar on the right-hand side of the plot) represents the order of α. If $\beta > 0$ and $L \to 0$, Figure 3 shows that α is roughly $O(k)$, which we discuss further in Corollary 6(a). In Figure 4, we visualize ω as a function of β and L, where k and U are fixed. The color represents the order of ω. In the dark blue region of the plot, Figure 4 shows that $\omega \to \infty$ when $b = \frac{2\beta}{L} \to k$, which provides insight into the extreme case for switching cost when $\beta \gtrsim \frac{kL}{2}$.
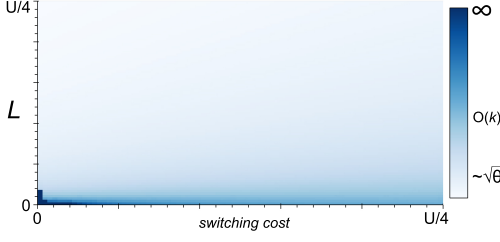
Fig. 3. DTPR-min: Plotting actual values of competitive ratio α for fixed $k \geq 1$, fixed $U > L$, and varying values for L and β (switching cost). Color represents the order of α for a given setting of θ and β.
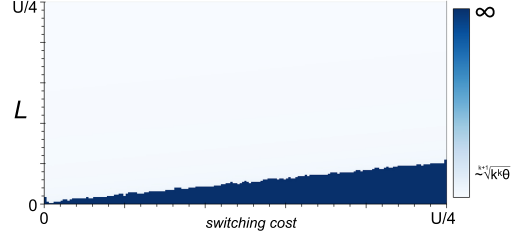

Fig. 4. DTPR-max: Plotting actual values of competitive ratio ω for fixed $k \geq 1$, fixed $U > L$, and varying values for L and β (switching cost). Color represents the order of ω for a given setting of θ and β.

To obtain additional insight into the form of the competitive ratios in Theorems 4 and 5, we present the following corollaries for two asymptotic regimes of interest: REGIME-1 captures the order of the competitive ratio when k is fixed and α or ω are sufficiently large, and REGIME-2 captures the order of the competitive ratio when $k \rightarrow \infty$.

COROLLARY 6. **(a)** *For* REGIME-1*, with fixed $k \geq 1$ and $\beta \in (0, \frac{U-L}{2})$, the competitive ratio of* DTPR-min *is*

$$\alpha \sim \frac{k\beta}{kL + 2\beta} + \sqrt{\frac{k^2 LU + 2kL\beta + 2kU\beta + 4\beta^2 + k^2\beta^2}{k^2 L^2 + 4kL\beta + 4\beta^2}}, \quad and \; \alpha \sim O(k) \; for \; L \rightarrow 0.$$

(b) *Furthermore, for* REGIME-2*, with $k \rightarrow \infty$ and $c = \frac{2\beta}{U}, c \in (0, \frac{U-L}{U})$, the competitive ratio of* DTPR-min *is*

$$\alpha \sim \left[W\left(\frac{\left(c + \frac{1}{\theta} - 1\right) e^c}{e} \right) - c + 1 \right]^{-1}.$$

COROLLARY 7. **(a)** *For* REGIME-1*, with fixed $k \geq 1$ and $b = \frac{2\beta}{L}, b \in (0, k)$, the competitive ratio of* DTPR-max *is*

$$\omega \sim O\left(\sqrt[k+1]{k^k \frac{k\theta}{k-b}} \right),$$

*and **(b)** for* REGIME-2*, with $k \rightarrow \infty$ and $b = \frac{2\beta}{L}, b \in (0, k)$, the competitive ratio of* DTPR-max *is*

$$\omega \sim W\left(\frac{\theta - 1 - b}{e^{1+b}} \right) + 1 + b.$$

Corollary 6(a) contextualizes the behavior of α (the competitive ratio of DTPR-min) in the most relevant OPR-min setting (when $\beta \in (0, \frac{U-L}{2})$). Note that in this minimization setting, as β grows, the competitive ratio improves. Let us also briefly discuss the other cases for the switching cost β, and why this interval makes sense. When $\beta > \frac{U-L}{2}$, the switching cost is large enough such that OPT only incurs a switching cost of 2β. In this regime, α does not fully capture the competitive ratio of DTPR-min, since every value in the threshold family $\{u_i\}_{i \in [1,k]}$ is at least U; in other words, whenever the algorithm begins accepting prices, it will accept k prices in a single continuous segment, incurring minimal switching cost of 2β. As $\beta \rightarrow \infty$, the competitive ratio of DTPR-min approaches 1. This theoretical result corresponds nicely with the empirical observation in [20] that a large switching overhead can nullify carbon emission reductions from temporal shifting if the job is interrupted frequently.

Conversely, Corollary 7(a) contextualizes the behavior of ω in the most relevant OPR-max setting (when $\beta \in (0, \frac{kL}{2})$), but we also discuss the other cases for the switching cost β, and why this interval makes sense. When $\beta \geq \frac{kL}{2}$, the switching cost is too large, and the competitive ratio may become unbounded. Note that this is shown explicitly in Figure 4. Consider an adversarial sequence which forces any OPR-max algorithm to accept k prices with value L at the end of the sequence. On such a sequence, even a player which incurs the minimum switching cost of 2β achieves zero or negative profit of $kL - 2\beta \leq 0$, and this is not well-defined.

Next, to begin to investigate the tightness of Theorems 4 and 5, it is interesting to consider special cases that correspond to models studied in previous work. In particular, when $\beta = 0$, i.e., there is no switching cost, OPR degenerates to the k-search problem [29]. For fixed $k \geq 1$ and $\theta \to \infty$, the optimal competitive ratios shown by [29] are $\sqrt{\theta/2}$ for k-min, and $\sqrt[k+1]{k^k\theta}$ for k-max (see Section 2.2).

Both versions of DTPR exactly recover the optimal k-search algorithms [29].[2] Figure 3 shows that if $\beta = 0$ and $L \to 0$, then $\alpha \to \infty$, which matches the k-min result of $\sqrt{\theta/2} \sim \infty$. Similarly, Figure 4 shows that if $\beta = 0$ and $L \to 0$, then $\omega \to \infty$, which matches the k-max result of $\sqrt[k+1]{k^k\theta} \sim \infty$.

More generally, one can ask if the competitive ratios of DTPR can be improved upon by other online algorithms outside of the special case of k-search. Our next set of results highlights that no improvement is possible, i.e., that DTPR-min and DTPR-max maintain the optimal competitive ratios possible for any deterministic online algorithm for OPR.

THEOREM 8. *Let $k \geq 1, \theta \geq 1$, and $\beta \in (0, \frac{U-L}{2})$. Then α given by Equation (9) is the best competitive ratio that a deterministic online algorithm for* OPR-min *can achieve.*

THEOREM 9. *Let $k \geq 1, \theta \geq 1$, and $\beta \in (0, \frac{kL}{2})$. Then ω given by Equation (10) is the best competitive ratio that a deterministic online algorithm for* OPR-max *can achieve.*

By combining Theorems 4 and 5 with Theorems 8 and 9, these results imply that the competitive ratios of DTPR-min and DTPR-max are optimal for OPR-min and OPR-max.

Finally, it is interesting to contrast the upper and lower bounds for OPR with those for k-search, since the contrast highlights the impact of switching costs. In OPR-min, when $\beta > 0$, we find that the DTPR-min competitive results *improve* on optimal results for k-min search (where $\beta = 0$ is assumed), particularly in the case where L approaches 0 (i.e., $\theta \to \infty$). Since Theorem 8 implies that DTPR-min is optimal, this shows that the addition of switching cost in OPR-min enables an online algorithm to achieve a better competitive ratio compared to k-min search, which is a surprising result. In contrast, for OPR-max with $\beta > 0$, DTPR-max's competitive bounds are *worse* than existing results for k-max search, particularly for large β. Since Theorem 9 implies that DTPR-max is optimal, this suggests that OPR-max is fundamentally a *more difficult* problem compared to k-max search.

We note that although the lower bounds shown in Theorems 8 and 9 specifically apply to deterministic algorithms, there are lower bounds in the literature for randomized k-search [29]. The randomized bound for k-min search are not an order-improvement over the deterministic lower bound, while the randomized results for k-max search improve the lower bound to $\Omega(\ln \theta)$. However, in the regimes of k which are interesting for applications (where k is sufficiently large), there will be a small difference between the deterministic upper bound and the randomized lower bound in practice. Combined, these results for k-search suggest that randomization similarly may

---

[2]To see this, note that by eliminating all β terms from Equations (9) and (10), we exactly recover Equations (3) and (4), which are the definitions of the k-search algorithms. When $\theta \to \infty$ as $L \to 0$, DTPR-min and DTPR-max match each k-search result exactly when $\beta = 0$. In Corollaries 6(b) and 7(b), DTPR-min and DTPR-max also match each k search result exactly when $k \to \infty$ and $\beta = 0$. (See Sec. 2.2)

not yield large improvements in the OPR setting. Exploring this dynamic further for OPR is an interesting direction for future work.

5 PROOFS

We now prove the results described in the previous section. In Section 5.1, we prove the DTPR-min results presented in Theorem 4 and Corollary 6. In Section 5.2, we provide a proof sketch for the lower bound results in Theorems 8 and 9, and defer the formal proofs to Appendix D. The competitive results for DTPR-max in Theorem 5 and Corollary 7 are deferred to Appendix B.

5.1 Competitive Results for DTPR-min

We begin by proving Theorem 4 and Corollary 6. The key novelty in the proof of the main competitive results (Theorems 4 and 5) lies in our effort to derive two threshold functions and balance the competitive ratio in several worst-case instances with respect to these thresholds, as outlined in Section 3.

PROOF OF THEOREM 4. For $0 \leq j \leq k$, let $\mathcal{S}_j \subseteq \mathcal{S}$ be the sets of OPR-min price sequences for which DTPR-min accepts exactly j prices (excluding the $k - j$ prices it is forced to accept at the end of the sequence). Then, all of the possible price sequences for OPR-min are represented by $\mathcal{S} = \bigcup_{j=0}^{k} \mathcal{S}_j$. Also, recall that by definition, $\ell_{k+1} = L$. Let $\epsilon > 0$ be a fixed constant, and define the following two price sequences σ_j and ρ_j:

$$\forall j \in [2, k] : \sigma_j = \ell_1, u_2, \dots, u_j, U, \underbrace{\ell_{j+1} + \epsilon, \dots, \ell_{j+1} + \epsilon}_{k}, \underbrace{U, U, \dots, U}_{k}.$$

$$\forall j \in [2, k] : \rho_j = \ell_1, U, \ell_2, U, \dots, U, \ell_j, U, \underbrace{\ell_{j+1} + \epsilon, \dots, \ell_{j+1} + \epsilon}_{k}, \underbrace{U, U, \dots, U}_{k}.$$

There are two special cases for $j = 0$ and $j = 1$. For $j = 0$, we have that $\sigma_0 = \rho_0$, and this sequence simply consists of $\ell_1 + \epsilon$ repeated k times, followed by U repeated k times. For $j = 1$, we also have that $\sigma_1 = \rho_1$, and this sequence consists of one price with value ℓ_1 and one price with value U, followed by $\ell_2 + \epsilon$ repeated k times and U repeated k times.

Observe that as $\epsilon \to 0$, σ_j and ρ_j are sequences yielding the worst-case ratios in $\mathcal{S}_j$, as DTPR-min is forced to accept $(k - j)$ worst-case U values at the end of the sequence, and each accepted value is exactly equal to the corresponding threshold.

Note that σ_j and ρ_j also represent two extreme possibilities for the additive switching cost. In σ_j, DTPR-min only switches twice, but it mostly accepts values u_i. In ρ_j, DTPR-min must switch $j + 1$ times because there are many intermediate U values, but it only accepts values ℓ_i.

In the worst case, we have

$$\frac{\text{DTPR-min}(\sigma_j)}{\text{OPT}(\sigma_j)} = \frac{\text{DTPR-min}(\rho_j)}{\text{OPT}(\rho_j)}.$$

Also, the optimal solutions for both sequences are lower bounded by the same quantity: $kc_{\min}(\sigma_j) + 2\beta = kc_{\min}(\rho_j) + 2\beta$. For any sequence s in $\mathcal{S}_j$, we have that $c_{\min}(s) > \ell_{j+1}$, so $\text{OPT}(\rho_j) = \text{OPT}(\sigma_j) \leq k\ell_{j+1} + 2\beta$.

By definition of the threshold families $\{\ell_i\}_{i \in [1,k]}$ and $\{u_i\}_{i \in [1,k]}$, we know that $\sum_{i=1}^{j} \ell_i + j2\beta = \sum_{i=1}^{j} u_i$ for any value $j \geq 2$:

$$\text{DTPR-min}(\rho_j) = \left(\sum_{i=1}^{j} \ell_i + (k - j)U + (j + 1)2\beta \right) = \left(\ell_1 + \sum_{i=2}^{j} u_i + (k - j)U + 4\beta \right) = \text{DTPR-min}(\sigma_j).$$

Note that whenever $j < 2$, we have that $\sigma_0 = \rho_0$, and $\sigma_1 = \rho_1$. Thus, $\text{DTPR-min}(\rho_j) = \text{DTPR-min}(\sigma_j)$ holds for any value of j. By definition of ℓ_1, we simplify $\ell_1 + \sum_{i=2}^{j} u_i + (k - j)U + 4\beta$ to $\sum_{i=1}^{j} u_i + (k - j)U + 2\beta$. Then, for any sequence $s \in \mathcal{S}_j$, we have the following:

$$\frac{\text{DTPR-min}(s)}{\text{OPT}(s)} \leq \frac{\text{DTPR-min}(\sigma_j)}{\text{OPT}(\sigma_j)} = \frac{\text{DTPR-min}(\rho_j)}{\text{OPT}(\rho_j)} \leq \frac{\sum_{i=1}^{j} u_i + (k - j)U + 2\beta}{k\ell_{j+1} + 2\beta}. \tag{11}$$

Before proceeding to the next step, we use an intermediate result stated in the following lemma with a proof given in Appendix C.

LEMMA 10. *For any $0 \leq j \leq k$, by definition of $\{\ell_i\}_{i \in [1,k]}$ and $\{u_i\}_{i \in [1,k]}$,*

$$\sum_{i=1}^{j} u_i + (k - j)U + 2\beta \leq \alpha \cdot (k\ell_{j+1} + 2\beta).$$

For $\epsilon \to 0$, the competitive ratio $\text{DTPR-min}/\text{OPT}$ is exactly α:

$$\forall 0 \leq j \leq k : \quad \frac{\text{DTPR-min}(\sigma_j)}{\text{OPT}(\sigma_j)} = \frac{\sum_{i=1}^{j} u_i + (k - j)U + 2\beta}{k\ell_{j+1} + 2\beta} = \alpha,$$

and thus for any sequence $s \in \mathcal{S}$,

$$\forall s \in \mathcal{S} : \quad \frac{\text{DTPR-min}(s)}{k c_{\min}(s) + 2\beta} \leq \alpha.$$

Since $\text{OPT}(s) \geq k c_{\min}(s) + 2\beta$ for any sequence s, this implies that DTPR-min is α-competitive. □

PROOF OF COROLLARY 6. To show part **(a)** for REGIME-1, with fixed $k \geq 1$, observe that we can expand the right-hand side of Equation (9) using the binomial theorem to obtain the following:

$$\frac{U - L - 2\beta}{U\left(1 - \frac{1}{\alpha}\right) - 2\beta\left(1 - \frac{1}{k} + \frac{1}{k\alpha}\right)} = 1 + \frac{1}{\alpha} + \Theta\left(\alpha^{-2}\right).$$

Next, observe that $\alpha^\star$ solving the following expression satisfies $\alpha^\star \geq \alpha \;\; \forall k : k \geq 1$, (i.e. $\alpha^\star$ is an upper bound of α):

$$\frac{U - L - 2\beta}{U\left(1 - \frac{1}{\alpha^\star}\right) - 2\beta\left(1 - \frac{1}{k} + \frac{1}{k\alpha^\star}\right)} = 1 + \frac{1}{\alpha^\star}.$$

By solving the above for $\alpha^\star$, we obtain

$$\alpha \sim \alpha^\star = \frac{k\beta}{kL + 2\beta} + \sqrt{\frac{k^2 LU + 2kL\beta + 2kU\beta + 4\beta^2 + k^2\beta^2}{k^2 L^2 + 4kL\beta + 4\beta^2}}.$$

Last, note that as $L \to 0$, we obtain the following result: $\alpha \sim \frac{k}{2} + \sqrt{\frac{kU}{2\beta} + 1 + \frac{k^2}{4}} \approx O(k)$.

To show part **(b)** for REGIME-2, we first observe that the right-hand side of Equation 9 can be approximated as $\left(1 + \frac{1}{k\alpha}\right)^k \approx e^{1/\alpha}$ when $k \to \infty$. Then by taking limits on both sides, we obtain the following:

$$\frac{U - L - 2\beta}{U\left(1 - \frac{1}{\alpha}\right) - 2\beta(1)} = e^{1/\alpha}.$$

For simplification purposes, let $\beta = cU/2$, where c is a small constant on the interval $\left(0, \frac{U-L}{U}\right)$. We then obtain the following:

$$\frac{U - L - cU}{U\left(1 - \frac{1}{\alpha}\right) - cU} = e^{1/\alpha} \implies L/U + c - 1 = \left(\frac{1}{\alpha} + c - 1\right)e^{1/\alpha}.$$

By definition of Lambert W function, solving this equation for α obtains the result in Corollary 6(b).
$\square$

5.2 Lower Bound Analysis: Proof of Theorem 8 (`OPR-min` Lower Bound)

In Theorems 8 and 9, we state that *any* deterministic strategy achieves a competitive ratio of at least α for `OPR-min`, and at least ω for `OPR-max`. In this section, we formalize the lower bound construction which proves Theorem 8. A similar construction is used to prove Theorem 9 in Appendix D.1. These two results jointly imply that our proposed DTPR algorithms are both optimal.

PROOF OF THEOREM 8. Let ALG be a deterministic online algorithm for `OPR-min`, and suppose that the adversary uses the price sequence $\ell_1, \ldots, \ell_k$, which is exactly the sequence defined by (5). ℓ_1 is presented to ALG, at most k times or until ALG accepts it. If ALG never accepts ℓ_1, the remainder of the sequence is all U, and ALG achieves a competitive ratio of $\frac{kU+2\beta}{k\ell_1+2\beta} = \alpha$, as defined in (7).

If ALG accepts ℓ_1, the next price presented is U, repeated at most k times *or until* ALG *switches to reject* U. After ALG has switched, ℓ_2 is presented to ALG, at most k times or until ALG accepts it. Again, if ALG never accepts ℓ_2, the remainder of the sequence is all U, and ALG achieves a competitive ratio of at least $\frac{\ell_1+(k-1)U+4\beta}{k\ell_2+2\beta} = \alpha$, as defined in (7).

As the sequence continues, whenever ALG does not accept some ℓ_i after it is presented k times, the adversary increases the price to U for the remainder of the sequence. Otherwise, if ALG accepts k prices before the end of the sequence, the adversary concludes by presenting L at least k times.

Observe that any ALG which does not immediately reject the first U presented to it after accepting some ℓ_i obtains a competitive ratio strictly worse than α. To illustrate this, suppose ALG has just accepted ℓ_1, incurring a cost of $\ell_1 + \beta$ so far. The adversary begins to present U, and ALG accepts $y \le (k-1)$ of these U prices before switching away. If $y = (k-1)$, ALG will accept k prices before the end of the sequence and achieve a competitive ratio of $\frac{\ell_1+(k-1)U+2\beta}{kL+2\beta} > \alpha$. Otherwise, if $y < (k-1)$, the cost incurred by ALG so far is at least $\ell_1 + 2\beta + yU$, while the cost incurred by ALG if it had immediately switched away ($y = 0$) would be $\ell_1 + 2\beta$ – since any price which might be accepted by ALG in the future should be $\le U$, the latter case strictly improves the competitive ratio of ALG.

Assuming that ALG does immediately reject any U presented to it, and that ALG accepts some prices before the end of the sequence, the competitive ratio attained by ALG is at least $\frac{\sum_{i=1}^{j}\ell_i+(j+1)2\beta+(k-j)U}{k\ell_{j+1}+2\beta} = \alpha$, as defined in (7).

Similarly, if ALG accepts k prices before the end of the sequence, the competitive ratio attained by ALG is at least $\frac{\sum_{i=1}^{k}\ell_i+k2\beta}{kL+2\beta} = \alpha$, as defined in (7).

Since any arbitrary deterministic online algorithm ALG cannot achieve a competitive ratio better than α playing against this adaptive adversary, our proposed algorithm DTPR-min is optimal. $\square$

6   CASE STUDY: CARBON-AWARE TEMPORAL WORKLOAD SHIFTING

We now present experimental results for the DTPR algorithms in the context of the carbon-aware temporal workload shifting problem. We evaluate DTPR-min (and DTPR-max in Appendix A) as compared to existing algorithms from the literature that have been adapted for OPR.

Table 2. Summary of carbon trace data sets

Location	🇺🇸 **Pacific NW, U.S.**	🇳🇿 **New Zealand**	🇨🇦 **Ontario, Canada**
Number of Data Points	10,144	1,324	17,898
Max. Carbon Intensity (U)	648 gCO$_2$eq/kWh	165 gCO$_2$eq/kWh	181 gCO$_2$eq/kWh
Min. Carbon Intensity (L)	18 gCO$_2$eq/kWh	54 gCO$_2$eq/kWh	15 gCO$_2$eq/kWh
Duration (mm/dd/yy)	04/20/22 - 12/06/22	10/19/21 - 11/16/21	10/19/21 - 12/06/22

6.1 Experimental Setup

We consider a carbon-aware load shifting system that operates on a hypothetical data center. An algorithm is given a deferrable and interruptible job that takes k time slots to complete, along with a deadline $T \geq k$, such that the job must be completed at most T slots after its arrival. The objective is to selectively run units of the job such that the total carbon emissions are minimized while still completing the job before its deadline.

For the minimization variant (OPR-min) of the experiments, we consider *carbon emissions inten-sities*, as the price values. At each time step t, the electricity supply has a carbon intensity c_t, i.e., if the job is being processed during the time step t ($x_t = 1$), the data center's carbon emissions during that time step are proportional to c_t. If the job is *not* being processed during the time step t ($x_t = 0$), we assume for simplicity that carbon emissions in the idle state are negligible and essentially 0. To model the combined computational overhead of interrupting, checkpointing, and restarting the job, the algorithm incurs a fixed switching cost of β whenever $x_{t-1} \neq x_t$, whose values are selected relative to the price values.

Carbon data traces. We use real-world carbon traces from Electricity Maps [32], which provide time-series information about the *average carbon emissions intensity* of the electric grid. We use traces from three different regions: the Pacific Northwest of the U.S., New Zealand, and Ontario, Canada. The data is provided at an hourly granularity and includes the current average carbon emissions intensity in grams of CO$_2$ equivalent per kilowatt-hour (gCO$_2$eq/kWh), and the percent-age of electricity being supplied from carbon-free sources. In Figure 9 (in Appendix A), we plot three representative actual traces for carbon intensity over time for a 96-hour period in each region.

Parameter settings. We test for time horizons (T) of 48 hours, 72 hours, and 96 hours. The chosen time horizon represents the time at which the job with length k must be completed. As is given in the carbon trace data, we consider time slots of one hour.

The online algorithms we use in experiments take L and U as parameters for their threshold functions. To set these parameters, we examine the entire carbon trace for the current location. For the Pacific NW trace and the Ontario trace, these values represent lower and upper bounds of the carbon intensity values for a full year. For the New Zealand trace, these values are a lower and upper bound for the values during a month of data, which is reflected by a smaller fluctuation ratio. We set L and U to be the minimum and maximum observed carbon intensity over the entire trace.

To generate each input sequence, a contiguous segment of size T is randomly sampled from the given carbon trace. In a few experiments, we simulate greater *volatility* over time by "scaling up" each price's deviation from the mean. First, we compute the average value over the entire sequence. Next, we compute the difference between each price and this average. Each of these differences is scaled by a *noise factor* of $m \geq 1$. Finally, new carbon values are computed by summing each scaled difference with the average. If $m = 1$, we recover the same sequence, and if $m > 1$, any deviation from the mean is proportionately amplified. Any values which become negative after applying this transformation are truncated to 0. This technique allows us to evaluate algorithms under different levels of volatility. As we are in the regime where $L = 0$, none of the other online algorithms considered have competitive guarantees, since their competitive ratios become unbounded when

Table 3. Summary of algorithms tested in our experiments

Algorithm	Carbon-aware	Switching-aware	Description
OPT (offline)	YES	YES	Optimal offline solution
Carbon-Agnostic	NO	YES	Runs job in the first k time slots
Const. Threshold	YES	NO	Runs job if carbon meets threshold $\sqrt{UL}$ [15]
k-search	YES	NO	Runs ith slot of job if carbon meets threshold Φ_i [29]
DTPR	YES	YES	This work (algorithms proposed in Section 3)

$L \to 0$. Instead, our DTPR algorithm maintains its optimal bound defined in (9) and (10) due to the presence of switching cost β in the competitive bounds. Performance in the presence of greater carbon volatility is important, as on-site renewable generation is seeing greater adoption as a supplementary power source for data centers [1, 36].

Benchmark algorithms. To evaluate the performance of DTPR, we use a dynamic programming approach to calculate the offline optimal solution for each given sequence and objective, which allows us to report the empirical competitive ratio for each tested algorithm. We compare DTPR against two categories of benchmark algorithms, which are summarized in Table 3.

The first category of benchmark algorithms is *carbon-agnostic* algorithms, which run the jobs during the first k time slots in order, i.e., accepting prices $c_1, \ldots, c_k$. This approach incurs the minimal switching cost of 2β, because it does not interrupt the job while it is being processed. The carbon-agnostic approach simulates the behavior of a scheduler that runs the job to completion as soon as it is submitted, without any focus on reducing carbon emissions. Note that the performance of this approach significantly varies based on the randomly selected sequence, since it will perform well if low-carbon electricity is available in the first few slots, and will perform poorly if the first few slots are high-carbon.

We also compare DTPR against *switching-cost-agnostic* algorithms, which only consider carbon cost. We have two algorithms of this type, each drawing from existing online search methods in the literature. Although they do not consider the switching cost in their design, they still incur a switching cost whenever their decision in adjacent time slots differs.

The first such algorithm is a *constant threshold algorithm*, which uses the $\sqrt{UL}$ threshold value first presented for online search in [15]. In our minimization experiments, this algorithm runs the workload during the first k time slots where the carbon intensity is at most $\sqrt{UL}$.

The other switching-cost-agnostic algorithm tested is the k-search algorithm shown by [29] and described in Section 2.2. The k-min search algorithm chooses to run the ith hour of the job during the first time slot where the carbon intensity is at most Φ_i.

6.2 Experimental Results

We now present our experimental results. Our focus is on the empirical competitive ratio (a lower competitive ratio is better). We report the performance of all algorithms for each experimental setting, in each tested region. Throughout the minimization experiments, we observe that DTPR-min outperforms the benchmark algorithms. The 95th percentile worst-case empirical competitive ratio achieved by DTPR-min is a 48.2% improvement on the carbon-agnostic method, a 15.6% improvement on the k-min search algorithm, and a 14.4% improvement on the constant threshold algorithm.

In Figure 5, we show results for three different values of horizon T in each carbon trace, with fixed $\beta \approx U/20$, fixed $k = \lceil T/6 \rceil$, and no added volatility. Although our experiments test three distinct values for T, we later observe that the *ratio between k and T* is the primary factor that changes the performance of the algorithms we test; in this figure, DTPR and the benchmark algorithms compare
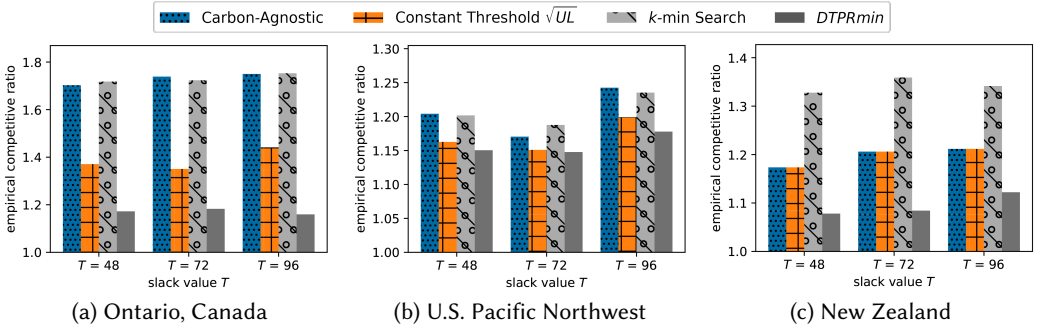
Fig. 5. Experiments for three distinct *time horizons*, where $T \in \{48, 72, 96\}$.
(a): Ontario, Canada carbon trace, with $\theta = 12.0\bar{6}$ (b): U.S. Pacific Northwest carbon trace, with $\theta = 36$
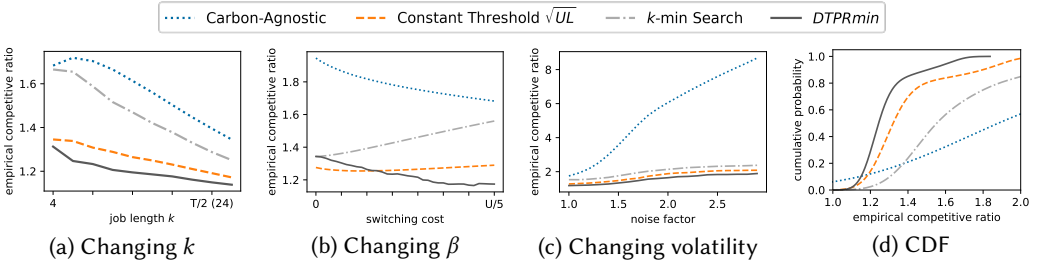(c): New Zealand carbon trace, with $\theta = 3.0\bar{5}$

Fig. 6. Experiments on Ontario, Canada carbon trace, with $\theta = 12.0\bar{6}$, and $T = 48$.
(a): Changing job length k w.r.t. time horizon T (x-axis), vs. competitive ratio (b): Changing switching cost β
w.r.t. U (x-axis), vs. competitive ratio (c): Different volatility levels w.r.t. U (x-axis), vs. competitive ratio
(d): Cumulative distribution function of competitive ratios

very similarly on the same carbon trace for different T values. As such, we set $T = 48$ in the rest of
the experiments in this section for brevity. This represents a *time horizon* of 48 hours.

In the first experiment, we test all algorithms for different job lengths k in the range from 4 hours
to $T/2$ (24 hours). The switching cost β is non-zero and fixed to $\approx U/20$, and no volatility is added to
the carbon trace. By testing different values for k, this experiment tests different ratios between the
workload length and the horizon provided to the algorithm. In Figures 6(a), 7(a), and 8(a), we show
that the competitive ratio of DTPR-min outperforms others, and it compares particularly favorably
for *short* job lengths. Averaging over all regions and job lengths, the competitive ratio achieved by
DTPR-min is a 11.4% improvement on the carbon-agnostic method, a 14.0% improvement on the
k-min search algorithm, and a 5.5% improvement on the constant threshold algorithm.

In the second experiment, we test all algorithms for different switching costs β in the range
from 0 to $U/5$. The job length k is set to 10 hours, and no volatility is added to the carbon trace.
By testing different values for β, this experiment tests how an increasing switching cost impacts
the performance of DTPR-min with respect to other algorithms which do not explicitly consider
the switching cost. In Figures 6(b), 7(b), and 8(b), we show that the observed competitive ratio of
DTPR-min outperforms the benchmark algorithms for most values of β in all regions. Unsurprisingly,
the carbon-agnostic technique (which incurs minimal switching cost) performs better as β grows.
While the constant threshold algorithm has relatively consistent performance, the k-min search
algorithm performs noticeably worse as β grows. Averaging over all regions and switching cost
values, the competitive ratio achieved by DTPR-min is a 18.2% improvement on the carbon-agnostic
method, a 8.9% improvement on the k-min search algorithm, and a 4.1% improvement on the
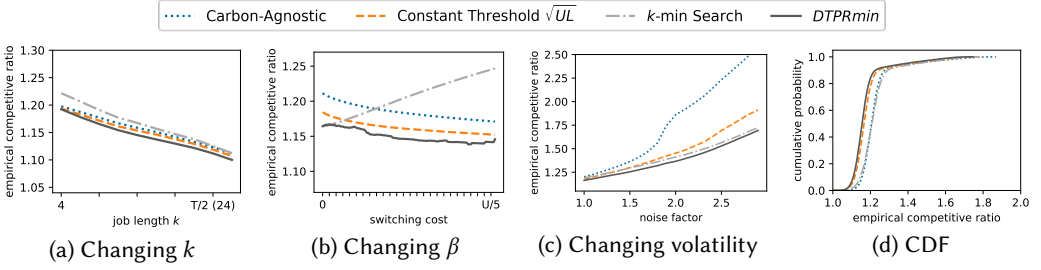constant threshold algorithm.

Fig. 7. Experiments on U.S. Pacific Northwest carbon trace, with $\theta = 36$, and $T = 48$.
(a): Changing job length k w.r.t. time horizon T (x-axis), vs. competitive ratio (b): Changing switching cost β w.r.t. U (x-axis), vs. competitive ratio (c): Different volatility levels w.r.t. U (x-axis), vs. competitive ratio (d): Cumulative distribution function of competitive ratios
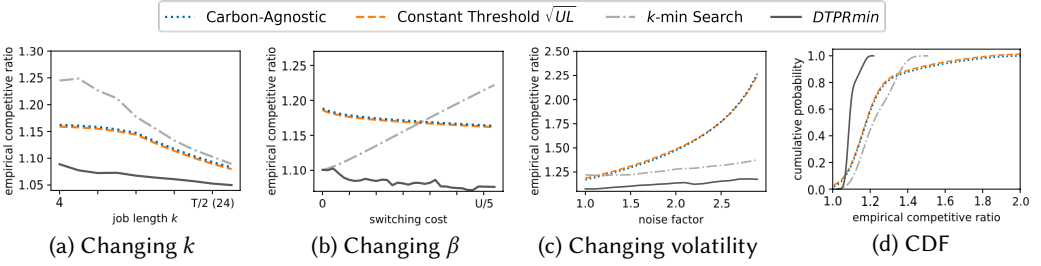

Fig. 8. Experiments on New Zealand carbon trace, with $\theta = 3.0\bar{5}$, and $T = 48$.
Note: the line for Carbon-Agnostic overlaps the line for Constant Threshold in some of the above plots.
(a): Changing job length k w.r.t. time horizon T (x-axis), vs. competitive ratio (b): Changing switching cost β w.r.t. U (x-axis), vs. competitive ratio (c): Different volatility levels w.r.t. U (x-axis), vs. competitive ratio (d): Cumulative distribution function of competitive ratios

In the final experiment, we test all algorithms on sequences with different volatility. The job length k and switching cost β are both fixed as previously. We add volatility by setting a *noise factor* from the range 1.0 to 3.0. By testing different values for this volatility, this experiment tests how each algorithm handles larger fluctuations in the carbon intensity of consecutive time steps. In Figures 6(c), 7(c), and 8(c), we show that the observed competitive ratio of DTPR-min outperforms the benchmark algorithms for all noise factors in all regions. Intuitively, higher volatility values cause the online algorithms to perform worse in general. Averaging over all regions and noise factors, the competitive ratio achieved by DTPR-min is a 53.6% improvement on the carbon-agnostic method, a 13.5% improvement on the k-min search algorithm, and a 14.3% improvement on the constant threshold algorithm.

By averaging over all experiments for a given region, we obtain the cumulative distribution function plot for each algorithm's competitive ratio in Figures 6(d), 7(d), and 8(d). Compared to the carbon-agnostic, constant threshold, and k-min search algorithms, DTPR-min achieves a lower average empirical competitive ratio distribution for all tested regions. Across *all regions* at the 95th percentile, DTPR-min achieves a worst-case empirical competitive ratio of 1.40. This represents a 48.2% improvement over the *carbon-agnostic* algorithm, and improvements of 15.6% and 14.4% over the k-min search and constant threshold *switching-cost-agnostic* algorithms, respectively.

7 RELATED WORK

This paper contributes directly to three lines of work: (i) work on online search and related problems, e.g., k-search, one-way trading, and online knapsack; (ii) work on online optimization problems

with switching costs, e.g., metrical task systems and convex function chasing; and (iii) work on carbon-aware load shifting. We describe the relationship to each below.

Online Search. The OPR problem is related to the online k-search problem [23, 29], as discussed in the introduction and Section 2.2. It also has several similar counterparts, including online conversion problems such as one-way trading [14, 15, 34, 44] and online knapsack problems [45, 50, 54], with practical applications to stock trading [29], cloud pricing [53], electric vehicle charging [46], etc. The k-search problem can be viewed as an integral version of the online conversion problem, while the general online conversion problem allows continuous one-way trading. The basic online knapsack problem studies how to pack arriving items of different sizes and values into a knapsack with limited capacity, while its extensions to item departures [45, 53] and multidimensional capacity [50] have also been studied recently. Another line of research leverages ML predictions to design learning-augmented online algorithms for online k-search [23] and online conversion [45]. However, to the best of our knowledge, none of these works consider the switching cost of changing decisions.

Metrical Task Systems. The metrical task systems (MTS) problem was introduced by Borodin et al. in [7]. Several decades of progress on upper and lower bounds on the competitive ratio of MTS recently culminated with a tight bound of $\Theta(\log^2 n)$ for the competitive ratio of MTS on an arbitrary n-point metric space, with $\Theta(\log n)$ being possible on certain metric spaces such as trees [8, 9]. Several modified forms of MTS have also seen significant attention in the literature, such as smoothed online convex optimization (SOCO) and convex function chasing (CFC), in which the decision space is an n-dimensional normed vector space and cost functions are restricted to be convex [17, 25]. The best known upper and lower bounds on the competitive ratio of CFC are $O(n)$ and $\Omega(\sqrt{n})$, respectively, in n-dimensional Euclidean spaces [10, 39]. However, algorithms with competitive ratios independent of dimension can be obtained for certain special classes of functions, such as α-polyhedral functions [11]. Several recent works have also investigated the design of learning-augmented algorithms for various cases of CFC/SOCO and MTS which exploit the performance of ML predictions of the optimal decisions [4, 12, 13, 24, 38]. The key characteristic distinguishing OPR from MTS is the presence of a deadline constraint. None of the algorithms for MTS-like problems are designed to handle long-term constraints while being competitive.

Carbon-Aware Temporal Workload Shifting. The goal of shifting workloads in time to allow more sustainable operations of data centers has been of interest for more than a decade, e.g., [19, 26–28]. Traditionally, such papers have used models that build on one of convex function chasing, k-search, or online knapsack to design algorithms; however such models do not capture both the switching costs and long-term deadlines that are crucial to practical deployment. In recent years, the load shifting literature has focused specifically on reducing the carbon footprint of operations, e.g., [1, 6, 36, 47]. Perhaps most related to this paper is [47], which explores the problem of carbon-aware temporal workload shifting and proposes a threshold-based algorithm that suspends the job when the carbon intensity is higher than a threshold value and resumes it when it drops below the threshold. However, it does not consider switching nor does it provide any deadline guarantees. Other recent work on carbon-aware temporal shifting seeks to address the resultant increase in job completion times. In [42], authors leverage the pause and resume approach to reduce the carbon footprint of ML training and high-performance computing applications such as BLAST [16]. However, instead of resuming at normal speed (1×) during the low carbon intensity periods, their applications resume operation at a faster speed (m×), where the scale factor m depends on the application characteristics. It uses a threshold-based approach to determine the low carbon intensity periods but does not consider switching costs or provide any deadline guarantees. A future direction is to extend the DTPR algorithms to consider the ability to scale up speed after resuming jobs.

In addition to our direct contributions in the above fields, our work is adjacent to several existing studies which have considered switching costs and *hysteretic control* in queueing models for single servers, server farms, and clouds. In [22], an M/M/1 queueing system is presented where the decision maker chooses arrival and service rates at each epoch and incurs a switching cost to change the rates. In this regime, they show that the optimal policy is a hysteretic policy, which exhibits resistance to change due to the switching cost. Gandhi et al. [18] present an M/M/k queueing system for server farms with setup costs, where turning a server on incurs a time delay. Similarly, [33] presents and analyzes a nearly-optimal mechanism to control the performance and power consumption of a server farm, where the setup cost incurs time and energy. A few works have also considered similar problems with different assumptions, such as job arrivals distributed according to a stochastic fluid model [5], and modeling the control policy as a Markov decision process [51]. It is notable that nearly all of these works derive hysteretic control policies based on the queue length, which essentially use a double threshold technique to resist changing decisions as a function of switching cost, a similar flavor of the result as we present in our setting. However, OPR is foundationally different as compared to the above works since we consider a single workload, a single deadline, and costs are exogenous to the online decision; this results in an algorithm design and analysis technique that differ substantially from these queueing models.

8 CONCLUDING REMARKS

Motivated by carbon-aware load shifting, we introduce and study the online pause and resume problem (OPR), which bridges gaps between several online optimization problems. To our knowledge, it is the first online optimization problem that includes both long-term constraints and switching costs. Our main results provide optimal online algorithms for the minimization and maximization variants of this problem, as well as lower bounds for the competitive ratio of any deterministic online algorithm. Notably, our proposed algorithms match existing optimal results for the related k-search problem when the switching cost is 0, and improve on the k-min search competitive bounds for non-zero switching cost. The key to our results is a novel double threshold algorithm that we expect to be applicable in other online problems with switching costs.

There are a number of interesting directions in which to continue the study of OPR. We have highlighted the application of OPR to carbon-aware load shifting, but OPR also applies to many other problems where pricing changes over time and frequent switching is undesirable. Pursuing these applications is important. Theoretically, there are several interesting open questions. First, considering the target application of carbon-aware load shifting, some workloads are *highly parallelizable* [42], which adds another dimension of scaling to the problem (i.e., instead of choosing to run 1 unit of the job in each time slot, the online player must decide how many units to allocate at each time slot). Furthermore, considering *heterogeneous switching costs* would be a logical extension of the setting we have considered here, modeling, for example, switching models which act as a function of the time spent in the current state. Both of these make the theoretical problem more challenging, and are important considerations for future work. Additionally, very recent work has incorporated machine-learned advice to achieve better performance on related online problems, including k-search [23, 44], CFC/SOCO [12, 24], and MTS [4, 13, 38]. Designing learning-augmented algorithms for OPR is a very promising line of future work, particularly considering applications such as carbon-aware load shifting, where accurate predictions can significantly improve the algorithm's understanding of the future in the best case, without sacrificing worst-case guarantees.

ACKNOWLEDGMENTS

DISCLAIMERS

REFERENCES

[1] Bilge Acun, Benjamin Lee, Fiodar Kazhamiaka, Kiwan Maeng, Udit Gupta, Manoj Chakkaravarthy, David Brooks, and Carole-Jean Wu. 2023. Carbon Explorer: A Holistic Framework for Designing Carbon Aware Datacenters. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2* (Vancouver, BC, Canada) *(ASPLOS 2023)*. Association for Computing Machinery, New York, NY, USA, 118–132. https://doi.org/10.1145/3575693.3575754

[2] Pradeep Ambati, Noman Bashir, David Irwin, Mohammad Hajiesmaili, and Prashant Shenoy. 2020. Hedge Your Bets: Optimizing Long-term Cloud Costs by Mixing VM Purchasing Options. In *2020 IEEE International Conference on Cloud Engineering (IC2E)*. 105–115. https://doi.org/10.1109/IC2E48712.2020.00018

[3] Spyros Angelopoulos, Christoph Dürr, Shendan Jin, Shahin Kamali, and Marc Renault. 2022. Online Computation with Untrusted Advice. arXiv:1905.05655 [cs.DS]

[4] Antonios Antoniadis, Christian Coester, Marek Elias, Adam Polak, and Bertrand Simon. 2020. Online Metric Algorithms with Untrusted Predictions. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 345–355.

[5] Naser M. Asghari, M. Mandjes, and Anwar Walid. 2014. Energy-efficient scheduling in multi-core servers. *Computer Networks* 59 (2014), 33–43. https://doi.org/10.1016/j.bjp.2013.12.009

[6] Noman Bashir, Tian Guo, Mohammad Hajiesmaili, David Irwin, Prashant Shenoy, Ramesh Sitaraman, Abel Souza, and Adam Wierman. 2021. Enabling Sustainable Clouds: The Case for Virtualizing the Energy System. In *Proceedings of the ACM Symposium on Cloud Computing* (Seattle, WA, USA) *(SoCC '21)*. Association for Computing Machinery, New York, NY, USA, 350–358. https://doi.org/10.1145/3472883.3487009

[7] Allan Borodin, Nathan Linial, and Michael E. Saks. 1992. An Optimal On-Line Algorithm for Metrical Task System. *J. ACM* 39, 4 (Oct 1992), 745–763. https://doi.org/10.1145/146585.146588

[8] Sébastien Bubeck, Christian Coester, and Yuval Rabani. 2023. The Randomized k-Server Conjecture Is False!. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing (STOC 2023)* (Orlando, FL, USA) *(STOC 2023)*. Association for Computing Machinery, New York, NY, USA, 581–594. https://doi.org/10.1145/3564246.3585132

[9] Sébastien Bubeck, Michael B. Cohen, James R. Lee, and Yin Tat Lee. 2021. Metrical Task Systems on Trees via Mirror Descent and Unfair Gluing. *SIAM J. Comput.* 50, 3 (Jan. 2021), 909–923. https://doi.org/10.1137/19M1237879

[10] Sébastien Bubeck, Bo'az Klartag, Yin Tat Lee, Yuanzhi Li, and Mark Sellke. 2019. Chasing Nested Convex Bodies Nearly Optimally. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Society for Industrial and Applied Mathematics, 1496–1508. https://doi.org/10.1137/1.9781611975994.91

[11] NiangJun Chen, Gautam Goel, and Adam Wierman. 2018. Smoothed Online Convex Optimization in High Dimensions via Online Balanced Descent. In *Proceedings of the 31st Conference On Learning Theory*. PMLR, 1574–1594.

[12] Nicolas Christianson, Tinashe Handina, and Adam Wierman. 2022. Chasing Convex Bodies and Functions with Black-Box Advice. In *Proceedings of the 35th Conference on Learning Theory*, Vol. 178. PMLR, 867–908.

[13] Nicolas Christianson, Junxuan Shen, and Adam Wierman. 2023. Optimal robustness-consistency tradeoffs for learning-augmented metrical task systems. In *International Conference on Artificial Intelligence and Statistics*.

[14] Peter Damaschke, Phuong Hoai Ha, and Philippas Tsigas. 2007. Online Search with Time-Varying Price Bounds. *Algorithmica* 55, 4 (Dec. 2007), 619–642. https://doi.org/10.1007/s00453-007-9156-9

[15] R. El-Yaniv, A. Fiat, R. M. Karp, and G. Turpin. 2001. Optimal Search and One-Way Trading Online Algorithms. *Algorithmica* 30, 1 (May 2001), 101–139. https://doi.org/10.1007/s00453-001-0003-0

[16] National Center for Biotechnology Information. 2022. Basic Local Alignment Search Tool (BLAST). https://blast.ncbi.nlm.nih.gov.

[17] Joel Friedman and Nathan Linial. 1993. On convex body chasing. *Discrete & Computational Geometry* 9, 3 (March 1993), 293–321. https://doi.org/10.1007/bf02189324

[18] Anshul Gandhi, Mor Harchol-Balter, and Ivo Adan. 2010. Server farms with setup costs. *Performance Evaluation* 67, 11 (2010), 1123–1138. https://doi.org/10.1016/j.peva.2010.07.004 Performance 2010.

[19] Vani Gupta, Prashant Shenoy, and Ramesh K Sitaraman. 2019. Combining renewable solar and open air cooling for greening internet-scale distributed networks. In *Proceedings of the Tenth ACM International Conference on Future Energy Systems*. 303–314.

[20] Walid A. Hanafy, Roozbeh Bostandoost, Noman Bashir, David Irwin, Mohammad Hajiesmaili, and Prashant Shenoy. 2023. The War of the Efficiencies: Understanding the Tension between Carbon and Energy Optimization. In *Proceedings of the 2nd Workshop on Sustainable Computer Systems*. ACM. https://doi.org/10.1145/3604930.3605709

[21] Abdolhossein Hoorfar and Mehdi Hassani. 2008. Inequalities on the Lambert W function and hyperpower function. *Journal of Inequalities in Pure and Applied Mathematics* 9, 51 (Jan. 2008). Issue 2.

[22] M. Yu. Kitaev and Richard F. Serfozo. 1999. M/M/1 Queues with Switching Costs and Hysteretic Optimal Control. *Operations Research* 47, 2 (1999), 310–312. https://doi.org/10.1287/opre.47.2.310

[23] Russell Lee, Bo Sun, John C. S. Lui, and Mohammad Hajiesmaili. 2022. Pareto-Optimal Learning-Augmented Algorithms for Online k-Search Problems. arXiv:2211.06567 https://arxiv.org/abs/2211.06567

[24] Pengfei Li, Jianyi Yang, and Shaolei Ren. 2022. Expert-Calibrated Learning for Online Optimization with Switching Costs. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 6, 2 (May 2022), 1–35. https://doi.org/10.1145/3530894

[25] Minghong Lin, Zhenhua Liu, Adam Wierman, and Lachlan L. H. Andrew. 2012. Online algorithms for geographical load balancing. In *2012 International Green Computing Conference (IGCC)*. IEEE. https://doi.org/10.1109/igcc.2012.6322266

[26] Minghong Lin, Adam Wierman, Lachlan LH Andrew, and Eno Thereska. 2012. Dynamic right-sizing for power-proportional data centers. *IEEE/ACM Transactions on Networking* 21, 5 (2012), 1378–1391.

[27] 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 *Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE joint international conference on Measurement and Modeling of Computer Systems*. 175–186.

[28] Zhenhua Liu, Minghong Lin, Adam Wierman, Steven H Low, and Lachlan LH Andrew. 2011. Greening geographical load balancing. *ACM SIGMETRICS Performance Evaluation Review* 39, 1 (2011), 193–204.

[29] Julian Lorenz, Konstantinos Panagiotou, and Angelika Steger. 2008. Optimal Algorithms for k-Search with Application in Option Pricing. *Algorithmica* 55, 2 (Aug. 2008), 311–328. https://doi.org/10.1007/s00453-008-9217-8

[30] Thodoris Lykouris and Sergei Vassilvtiskii. 2018. Competitive Caching with Machine Learned Advice. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, 3296–3305. https://proceedings.mlr.press/v80/lykouris18a.html

[31] Diptyaroop Maji, Ramesh K. Sitaraman, and Prashant Shenoy. 2022. DACF: Day-Ahead Carbon Intensity Forecasting of Power Grids Using Machine Learning. In *Proceedings of the Thirteenth ACM International Conference on Future Energy Systems* (Virtual Event) *(e-Energy '22)*. Association for Computing Machinery, New York, NY, USA, 188–192. https://doi.org/10.1145/3538637.3538849

[32] Electricity Maps. 2020. Electricity Map. https://www.electricitymap.org/map.

[33] Isi Mitrani. 2011. Managing performance and power consumption in a server farm. *Annals of Operations Research* 202, 1 (July 2011), 121–134. https://doi.org/10.1007/s10479-011-0932-1

[34] Esther Mohr, Iftikhar Ahmad, and Günter Schmidt. 2014. Online algorithms for conversion problems: a survey. *Surveys in Operations Research and Management Science* 19, 2 (2014), 87–104.

[35] Manish Purohit, Zoya Svitkina, and Ravi Kumar. 2018. Improving Online Algorithms via ML Predictions. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc.

[36] Ana Radovanovic, Ross Koningstein, Ian Schneider, Bokan Chen, Alexandre Duarte, Binz Roy, Diyue Xiao, Maya Haridasan, Patrick Hung, Nick Care, et al. 2022. Carbon-Aware Computing for Datacenters. *IEEE Transactions on Power Systems* (2022).

[37] Samyam Rajbhandari, Olatunji Ruwase, Jeff Rasley, Shaden Smith, and Yuxiong He. 2021. ZeRO-Infinity: Breaking the GPU Memory Wall for Extreme Scale Deep Learning. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–14.

[38] Daan Rutten, Nicolas Christianson, Debankur Mukherjee, and Adam Wierman. 2022. Smoothed Online Optimization with Unreliable Predictions. https://doi.org/10.48550/arXiv.2202.03519

[39] Mark Sellke. 2020. Chasing Convex Bodies Optimally. In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '20)*. Society for Industrial and Applied Mathematics, USA, 1509–1518.

[40] Supreeth Shastri, Amr Rizk, and David Irwin. 2016. Transient guarantees: Maximizing the value of idle cloud capacity. In *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 992–1002.

[41] Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, et al. 2022. Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, a Large-Scale Generative Language Model. *arXiv preprint arXiv:2201.11990* (2022).

[42] Abel Souza, Noman Bashir, Jorge Murillo, Walid Hanafy, Qianlin Liang, David Irwin, and Prashant Shenoy. 2023. Ecovisor: A Virtual Energy System for Carbon-Efficient Applications. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2* (Vancouver, BC, Canada) *(ASPLOS 2023)*. Association for Computing Machinery, New York, NY, USA, 252–265. https://doi.org/10.1145/3575693.3575709

[43] Seán M. Stewart. 2009. On Certain Inequalities Involving the Lambert W function. *Journal of Inequalities in Pure and Applied Mathematics* 10, 96 (Nov. 2009). Issue 4.

[44] Bo Sun, Russell Lee, Mohammad Hajiesmaili, Adam Wierman, and Danny Tsang. 2021. Pareto-Optimal Learning-Augmented Algorithms for Online Conversion Problems. In *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34. Curran Associates, Inc., 10339–10350. https://proceedings.neurips.cc/paper_files/paper/2021/file/55a988dfb00a914717b3000a3374694c-Paper.pdf

[45] Bo Sun, Lin Yang, Mohammad Hajiesmaili, Adam Wierman, John CS Lui, Don Towsley, and Danny HK Tsang. 2022. The Online Knapsack Problem with Departures. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 6, 3 (2022), 1–32.

[46] Bo Sun, Ali Zeynali, Tongxin Li, Mohammad Hajiesmaili, Adam Wierman, and Danny HK Tsang. 2020. Competitive Algorithms for the Online Multiple Knapsack Problem With Application to Electric Vehicle Charging. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 4, 3 (2020), 1–32.

[47] Philipp Wiesner, Ilja Behnke, Dominik Scheinert, Kordian Gontarska, and Lauritz Thamsen. 2021. Let's Wait AWhile: How Temporal Workload Shifting Can Reduce Carbon Emissions in the Cloud. In *Proceedings of the 22nd International Middleware Conference*. Association for Computing Machinery, New York, NY, USA, 260–272.

[48] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models. In *International Conference on Machine Learning (Proceedings of Machine Learning Research)*. PMLR, PMLR, Honolulu, HI, USA, 38087–38099.

[49] Lin Yang, Mohammad H. Hajiesmaili, Ramesh Sitaraman, Adam Wierman, Enrique Mallada, and Wing S. Wong. 2020. Online Linear Optimization with Inventory Management Constraints. *Proc. ACM Meas. Anal. Comput. Syst.* 4, 1, Article 16 (may 2020), 29 pages. https://doi.org/10.1145/3379482

[50] Lin Yang, Ali Zeynali, Mohammad H. Hajiesmaili, Ramesh K. Sitaraman, and Don Towsley. 2021. Competitive Algorithms for Online Multidimensional Knapsack Problems. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 5, 3, Article 30 (Dec 2021), 30 pages.

[51] Zexi Yang, Meng-Hsi Chen, Zhisheng Niu, and Dawei Huang. 2011. An Optimal Hysteretic Control Policy for Energy Saving in Cloud Computing. In *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*. 1–5. https://doi.org/10.1109/GLOCOM.2011.6133628

[52] Sheng Shui Zhang. 2006. The effect of the charging protocol on the cycle life of a Li-ion battery. *Journal of Power Sources* 161, 2 (2006), 1385–1391. https://doi.org/10.1016/j.jpowsour.2006.06.040

[53] ZiJun Zhang, Zongpeng Li, and Chuan Wu. 2017. Optimal posted prices for online cloud resource allocation. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 1, 1 (2017), 1–26.

[54] Yunhong Zhou, Deeparnab Chakrabarty, and Rajan Lukose. 2008. Budget Constrained Bidding in Keyword Auctions and Online Knapsack Problems. In *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Heidelberg, DE, 566–576. https://doi.org/10.1007/978-3-540-92185-1_63
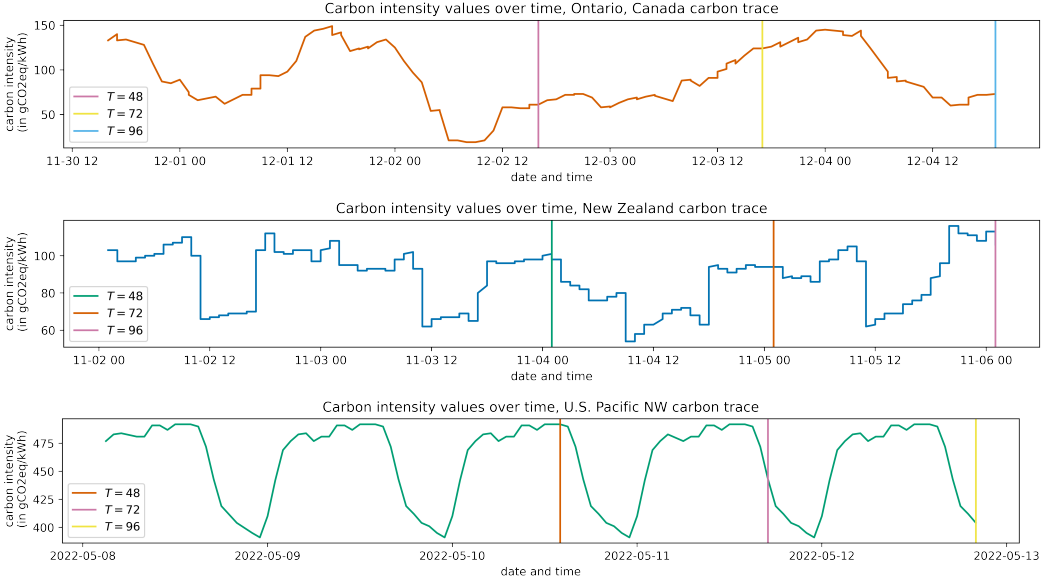
Fig. 9. Carbon intensity (in gCO$_2$eq/kWh) values plotted for each region tested in our numerical experiments, with one-hour granularity. We plot a representative random interval of 96 hours, with vertical lines demarcating the different values for T (time horizon) tested in our experiments. In all regions, carbon values roughly follow a diurnal (daily cycle) pattern. Actual values and observed intensities significantly vary in different regions.

A CASE STUDY RESULTS FOR DTPR-MAX ALGORITHM

Algorithm 2 Double Threshold Pause and Resume for OPR-max (DTPR-max)

Input: threshold values $\{u_i\}_{i \in [1,k]}$ and $\{\ell_i\}_{i \in [1,k]}$ defined in Equation (6), deadline T
Output: online decisions $\{x_t\}_{t \in [1,T]}$
  1: **initialize:** i = 1;
  2: **while** price c_t arrives **and** $i \leq k$ **do**
  3:     **if** $(k - i) \geq (T - t)$ **then**   ▷ close to the deadline T, we must accept remaining prices
  4:         price c_t is accepted, set $x_t = 1$
  5:     **else if** $x_{t-1} = 0$ **then**   ▷ If previous price was not accepted
  6:         **if** $c_t \geq u_i$ **then**   price c_t is accepted, set $x_t = 1$
  7:         **else** price c_t is rejected, set $x_t = 0$
  8:     **else if** $x_{t-1} = 1$ **then**   ▷ If previous price was accepted
  9:         **if** $c_t \geq \ell_i$ **then** price c_t is accepted, set $x_t = 1$
 10:         **else** price c_t is rejected, set $x_t = 0$
 11:     update $i = i + x_t$

This section presents and discusses the deferred experimental results for the DTPR-max algorithm (pseudocode summarized in Algorithm 2) in the carbon-aware temporal workload shifting case study. We evaluate DTPR-max against the same benchmark algorithms described in Section 6.1.

For the *maximization metric*, we consider the *percentage of carbon-free electricity* powering the grid. At each time step t, the electricity supply has a carbon-free percentage c_t, i.e., if the job is being processed during time slot t ($x_t = 1$), the electricity powering the data center's is c_t%
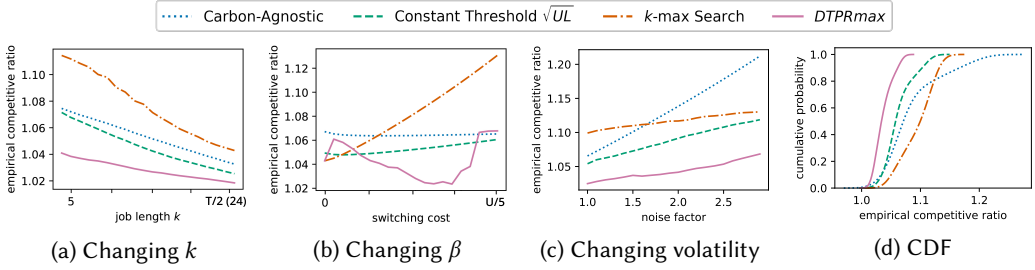
Fig. 10. Maximization experiments on Ontario, Canada carbon trace, with $\theta \approx 1.51$ and $T = 48$.
(a): Changing job length k w.r.t. time horizon T (x-axis), vs. competitive ratio (b): Changing switching cost β w.r.t. U (x-axis), vs. competitive ratio (c): Different volatility levels w.r.t. U (x-axis), vs. competitive ratio (d): Cumulative distribution function of competitive ratios

carbon-free, and the objective is to maximize this percentage over all k slots of the active running of the workload.

In these maximization experiments, the switching-cost-agnostic k-max-search algorithm chooses to run the ith hour of the job during the first time slot where the carbon-free supply is at least Φ_i. Similarly, the constant threshold algorithm chooses to run the job whenever the carbon-free supply is at least $\sqrt{UL}$. We set L and U to be the minimum and maximum carbon-free supply percentages over the entire trace being studied.

As in Section 6.2, our focus is on the competitive ratio (lower competitive ratio is better). We report the performance of all algorithms for each experiment setting, in each tested region.

In the first experiment, we test all algorithms for different job lengths k in the range from 4 hours to $T/2(24)$. The switching cost β is non-zero and fixed, and no volatility is added to the carbon trace. By testing different values for k, this experiment tests different ratios between the workload length and the slack provided to the algorithm. In Figures 10(a), 11(a), and 12(a), we show that the observed average competitive ratio of DTPR-max narrowly outperforms the benchmark algorithms for all values of k in all regions, and it compares particularly favorably for *short* job lengths. Averaging over all regions and job lengths, the competitive ratio achieved by DTPR-max is a 4.9% improvement on the carbon-agnostic method, a 8.4% improvement on the k-max search algorithm, and a 2.1% improvement on the constant threshold algorithm.

In the second experiment, we test all algorithms for different switching costs β in the range from 0 to $U/5$. The job length k is set to 10 hours, and no volatility is added to the carbon trace. By testing different values for β, this experiment tests how an increasing switching cost impacts the performance of DTPR-max with respect to other algorithms which do not explicitly consider the switching cost. In Figures 10(b), 11(b), and 12(b), we show that the average competitive ratio of DTPR-max notably outperforms the other algorithms for a wide range of β values in all regions. Unsurprisingly, the carbon-agnostic technique (which only incurs a switching cost of 2β) is more competitive as β grows. The k-max search algorithm performs noticeably worse as β grows. While the constant threshold algorithm has relatively consistent performance, the k-max search algorithm performs noticeably worse as β grows. Averaging over all regions and switching cost values, the competitive ratio achieved by DTPR-max is a 2.5% improvement on the carbon-agnostic method, a 6.4% improvement on the k-max search algorithm, and a 0.1% improvement on the constant threshold algorithm.

In the final experiment, we test all algorithms on sequences with different volatility. The job length k and switching cost β are both fixed. We add volatility by setting a *noise factor* from the range 1.0 to 3.0. By testing different values for this volatility, this experiment tests how each algorithm
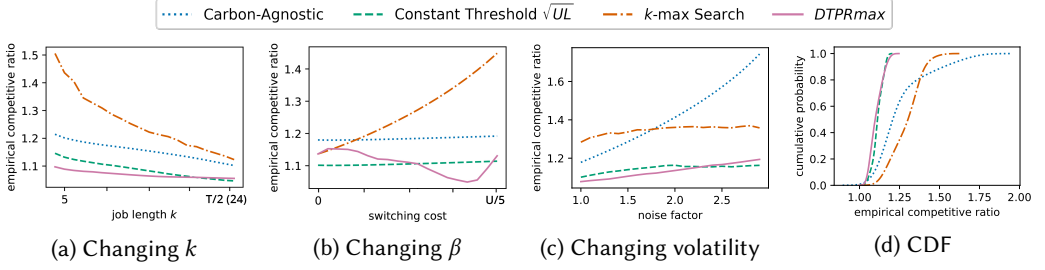
Fig. 11. Maximization experiments on U.S. Pacific Northwest carbon trace, with $\theta \approx 5.24$ and $T = 48$.
(a): Changing job length k w.r.t. time horizon T (x-axis), vs. competitive ratio (b): Changing switching cost β
w.r.t. U (x-axis), vs. competitive ratio (c): Different volatility levels w.r.t. U (x-axis), vs. competitive ratio (d):
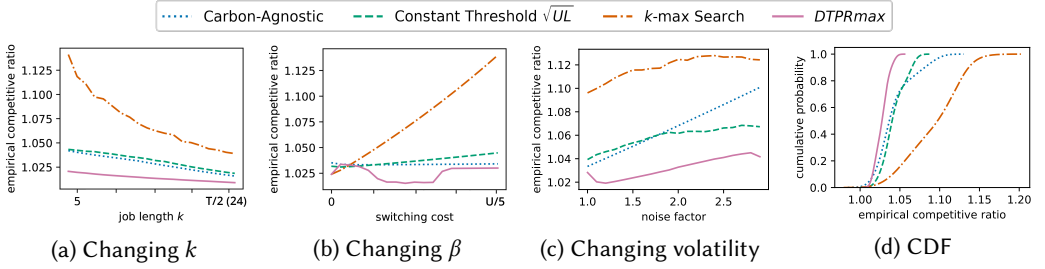Cumulative distribution function of competitive ratios

Fig. 12. Maximization experiments on New Zealand carbon trace, with $\theta \approx 1.35$ and $T = 48$.
(a): Changing job length k w.r.t. time horizon T (x-axis), vs. competitive ratio (b): Changing switching cost β
w.r.t. U (x-axis), vs. competitive ratio (c): Different volatility levels w.r.t. U (x-axis), vs. competitive ratio (d):
Cumulative distribution function of competitive ratios

handles larger fluctuations in the carbon intensity of consecutive time steps. In Figures 10(c), 11(c),
and 12(c), we show that the observed average competitive ratio of DTPR-max outperforms the other
algorithms for most noise factors in all regions, with a slight degradation in the Pacific Northwest
region. Intuitively, higher volatility values cause the online algorithms to perform worse in general.
Averaging over all regions and noise factors, the competitive ratio achieved by DTPR-max is a 13.0%
improvement on the carbon-agnostic method, a 11.2% improvement on the k-max search algorithm,
and a 2.1% improvement on the constant threshold algorithm.

By averaging over all experiments for a given region, we obtain the cumulative distribution
function plot for each algorithm's competitive ratio in Figures 10(d), 11(d), and 12(d). Compared
to the carbon-agnostic, constant threshold, and k-max search algorithms, DTPR-max generally
exhibits a lower average empirical competitive ratio over the tested regions. Notably, all of the
algorithms are nearly 1-competitive in our experiments. Compared to our minimization experiments,
DTPR-max outperforms the baseline algorithms by a smaller margin. Across *all regions* at the 95th
percentile, DTPR-max achieves a worst-case empirical competitive ratio of 1.08. This represents a
16.1% improvement over the *carbon-agnostic* algorithm, and improvements of 11.4% and 2.19% over
the k-max search and constant threshold *switching-cost-agnostic* algorithms, respectively.

We conjecture that one dynamic contributing to this is the relatively low values of θ observed
for the carbon-free supply percentage in these real-world carbon traces.

B COMPETITIVE ANALYSIS OF DTPR-MAX: PROOF OF THEOREM 5

Here we prove the DTPR-max results presented in Theorem 5 and Corollary 7.

PROOF OF THEOREM 5. For $0 \le j \le k$, let $\mathcal{S}_j \subseteq \mathcal{S}$ be the sets of OPR-max price sequences for which DTPR-max accepts exactly j prices (excluding the $k - j$ prices it is forced to accept at the end of the sequence). Then all of the possible price sequences for OPR-max are represented by $\mathcal{S} = \bigcup_{j=0}^{k} \mathcal{S}_j$. By definition, $u_{k+1} = U$. Let $\epsilon > 0$ be fixed, and define the following two price sequences σ_j and ρ_j:

$$\forall 0 \le j \le k : \sigma_j = u_1, \ell_2, \ldots, \ell_j, L, \underbrace{u_{j+1} - \epsilon, \ldots, u_{j+1} - \epsilon}_{k}, \underbrace{L, L, \ldots, L}_{k}.$$

$$\forall 0 \le j \le k : \rho_j = u_1, L, u_2, L, \ldots, L, u_j, L, \underbrace{u_{j+1} - \epsilon, \ldots, u_{j+1} - \epsilon}_{k}, \underbrace{L, L, \ldots, L}_{k}.$$

We have two special cases for $j = 0$ and $j = 1$. For $j = 0$, we have that $\sigma_0 = \rho_0$, and this sequence simply consists of $u_1 - \epsilon$ repeated k times, followed by L repeated k times. For $j = 1$, we also have that $\sigma_1 = \rho_1$, and this sequence consists of one price with value u_1 and one price with value L, followed by $u_2 - \epsilon$ repeated k times and L repeated k times.

Observe that as $\epsilon \to 0$, σ_j and ρ_j are sequences yielding the worst-case ratios in $\mathcal{S}_j$, as DTPR-max is forced to accept $(k - j)$ worst-case L values at the end of the sequence, and each accepted value is exactly equal to the corresponding threshold.

σ_j and ρ_j also represent two extreme possibilities for the switching cost. In σ_j, DTPR-max only switches twice, but it mostly accepts values ℓ_i. In ρ_j, DTPR-max must switch $j + 1$ times because there are many intermediate L values, but it only accepts values which are at least u_i.

Observe that $\text{OPT}(\sigma_j)/\text{DTPR-max}(\sigma_j) = \text{OPT}(\rho_j)/\text{DTPR-max}(\rho_j)$. First, the optimal solution for both sequences is exactly the same: $kc_{\max}(\sigma_j) - 2\beta = kc_{\max}(\rho_j) - 2\beta$.
For any sequence s in $\mathcal{S}_j$, we also know that $c_{\max}(s) < u_{j+1}$, so $\text{OPT}(\rho_j) = \text{OPT}(\sigma_j) \le ku_{j+1} - 2\beta$.

By definition of the threshold families $\{u_i\}_{i \in [1,k]}$ and $\{\ell_i\}_{i \in [1,k]}$, we know that $\sum_{i=1}^{j} u_i - j2\beta = \sum_{i=1}^{j} \ell_i$ for any value $j \ge 2$:

$$\text{DTPR-max}(\rho_j) = \left(u_1 + \sum_{i=2}^{j} \ell_i + (k - j)L - 4\beta \right) = \left(\sum_{i=1}^{j} u_i + (k - j)L - (j + 1)2\beta \right) = \text{DTPR-max}(\sigma_j).$$

Note that whenever $j < 2$, we have that $\sigma_0 = \rho_0$, and $\sigma_1 = \rho_1$. Thus, $\text{DTPR-min}(\rho_j) = \text{DTPR-min}(\sigma_j)$ holds for any value of j.

By definition of u_1, we simplify $u_1 + \sum_{i=2}^{j} \ell_i + (k - j)L - 4\beta$ to $\sum_{i=1}^{j} \ell_i + (k - j)L - 2\beta$. For any sequence $s \in \mathcal{S}_j$, we have the following:

$$\frac{\text{OPT}(s)}{\text{DTPR-max}(s)} \le \frac{\text{OPT}(\sigma_j)}{\text{DTPR-max}(\sigma_j)} = \frac{\text{OPT}(\rho_j)}{\text{DTPR-max}(\rho_j)} \le \frac{ku_{j+1} - 2\beta}{\sum_{i=1}^{j} \ell_i + (k - j)L - 2\beta}. \tag{12}$$

LEMMA 11. *For any $j \in [0, k]$, by definition of $\{u_i\}_{i \in [1,k]}$ and $\{\ell_i\}_{i \in [1,k]}$,*

$$\omega \cdot \left(\sum_{i=1}^{j} \ell_i + (k - j)L - 2\beta \right) \le ku_{j+1} - 2\beta. \qquad \textit{The proof is deferred to Appendix C.}$$

For $\epsilon \to 0$, the competitive ratio $\text{OPT}/\text{DTPR-max}$ is exactly ω:

$$\forall 0 \le j \le k : \quad \frac{\text{OPT}(\sigma_j)}{\text{DTPR-max}(\sigma_j)} = \frac{ku_{j+1} - 2\beta}{\sum_{i=1}^{j} \ell_i + (k - j)L - 2\beta} = \omega.$$

and thus for any sequence $s \in \mathcal{S}$,

$$\forall s \in \mathcal{S} : \quad \frac{k c_{\max}(s) - 2\beta}{\text{DTPR-max}(s)} \leq \omega.$$

Since $\text{OPT}(s) \leq k c_{\max}(s) - 2\beta$ for any sequence s, this implies that DTPR-max is ω-competitive. $\quad\square$

PROOF OF COROLLARY 7. For simplification purposes, let $\beta = bL/2$, where b is a real constant on the interval $(0, k)$. To show part **(a)** for REGIME-1, with fixed $k \geq 1$, observe that for sufficiently large ω, we have the following:

$$\theta - b - 1 = (\omega - 1) \left(1 + \frac{\omega}{k}\right)^k - \left(b - \frac{b}{k} + \frac{b\omega}{k}\right) \left(1 + \frac{\omega}{k}\right)^k \approx (1 + o(1)) \left[\omega \left(\frac{\omega}{k}\right)^k - b \left(\frac{\omega}{k}\right)^{k+1} - b\right].$$

Let $\omega_+ = \sqrt[k+1]{k^k \cdot \frac{k\theta}{k-b}}$. Then, for sufficiently large ω, we have the following:

$$(1 + o(1)) \left[\omega_+ \left(\frac{\omega_+}{k}\right)^k - b \left(\frac{\omega_+}{k}\right)^{k+1} - b\right] = (1 + o(1)) \frac{(k-b)(\theta)}{k - b} = (1 + o(1)) \left[\theta - b\right].$$

Furthermore, let $\varepsilon > 0$ and set $\omega_- = (1 - \varepsilon) \sqrt[k+1]{k^k \cdot \frac{k\theta}{k-b}}$.

A similar calculation as above shows that for sufficiently large θ we have:

$$(\omega_- - 1) \left(1 + \frac{\omega_-}{k}\right)^k - \left(b - \frac{b}{k} + \frac{a\omega_-}{k}\right) \left(1 + \frac{\omega_-}{k}\right)^k \geq (1 - 3k\varepsilon) \left[\theta - b\right].$$

Thus, $\omega = O\left(\sqrt[k+1]{k^k \frac{k\theta}{k-b}}\right)$ satisfies (10) for sufficiently large ω, fixed $k \geq 1$, and $\beta = \frac{bL}{2}$ s.t. $b \in (1, k)$.

To show part **(b)** for REGIME-2, observe that the right-hand side of (10) can be approximated as $\left(1 + \frac{\omega}{k}\right)^k \approx e^\omega$ when $k \to \infty$. Then by taking limits on both sides, we obtain the following:

$$\frac{U - L - 2\beta}{L(\omega - 1) - 2\beta(1)} = e^\omega.$$

Let $\beta = bL/2$ as outlined above. We then obtain the following:

$$\frac{U - L - bL}{L(\omega - 1) - bL} = \frac{\theta - 1 - b}{\omega - 1 - b} = e^\omega \implies \theta - 1 - b = (\omega - 1 - b) e^\omega.$$

By definition of the Lambert W function, solving this equation for ω obtains part (2). $\quad\square$

C PROOFS OF LEMMAS 10 AND 11

In this section, we give the deferred proofs of Lemmas 10 and 11, which are used in the proofs of Theorem 4 and Theorem 5, respectively.

PROOF OF LEMMA 10. We show that the following holds for any $j \in [0, k]$, by Definition 1:

$$\sum_{i=1}^{j} u_i + (k - j)U + 2\beta \leq \alpha \cdot (k\ell_{j+1} + 2\beta).$$

First, note that $k\ell_{j+1} = k(u_{j+1} - 2\beta)$ for all $j \in [0, k]$, by Observation 3. This gives us the following:

$$\sum_{i=1}^{j} u_i + (k-j)U + 2\beta \le \alpha k u_{j+1} + \alpha 2\beta - \alpha k 2\beta,$$

$$\sum_{i=1}^{j} u_i + (k-j)U + [2\beta - \alpha 2\beta + \alpha k 2\beta] \le \alpha k u_{j+1},$$

$$\frac{(k-j)U}{\alpha k} + \frac{\sum_{i=1}^{j} u_i}{\alpha k} + \left[\frac{2\beta}{\alpha k} - \frac{2\beta}{k} + 2\beta\right] \le u_{j+1}.$$

By substituting Def. 1 into $\sum_{i=1}^{j} u_i$, the above can be simplified exactly to the closed form for u_{j+1}:

$$\frac{U}{\alpha} - \frac{jU}{\alpha k} + \left(\frac{\sum_{i=1}^{j} u_i}{\alpha k}\right) + \left[\frac{2\beta}{\alpha k} - \frac{2\beta}{k} + 2\beta\right] = u_{j+1},$$

$$\left[U - \left(U - \frac{1}{\alpha}\right)\left(1 + \frac{1}{\alpha k}\right)^{j}\right] + \left[\left(\frac{2\beta}{\alpha k} - \frac{2\beta}{k} + 2\beta\right)\left(1 + \frac{1}{\alpha k}\right)^{j}\right] = u_{j+1}.$$

and the claim follows by the definition of u_{j+1}. □

PROOF OF LEMMA 11. We show that the following holds for any $j \in [0, k]$, by Definition 2:

$$\omega \cdot \left(\sum_{i=1}^{j} \ell_i + (k-j)L - 2\beta\right) \le k u_{j+1} - 2\beta.$$

First, note that $k u_{j+1} = k(\ell_{j+1} + 2\beta)$ for all $j \in [0, k]$, by Observation 3. This gives us the following:

$$\sum_{i=1}^{j} \ell_i + (k-j)L - 2\beta \le \frac{k\ell_{j+1}}{\omega} - \frac{2\beta}{\omega} + \frac{k2\beta}{\omega},$$

$$\sum_{i=1}^{j} \ell_i + (k-j)L - \left[2\beta - \frac{2\beta}{\omega} + \frac{k2\beta}{\omega}\right] \le \frac{k\ell_{j+1}}{\omega},$$

$$\frac{\omega\left(\sum_{i=1}^{j} \ell_i\right)}{k} + \frac{\omega(k-j)L}{k} - \left[\frac{\omega 2\beta}{k} - \frac{2\beta}{k} + 2\beta\right] \le \ell_{j+1}.$$

By substituting Def. 2 into $\sum_{i=1}^{j} \ell_i$, the above can be simplified exactly to the closed form for ℓ_{j+1}:

$$\omega L - \frac{\omega j L}{k} + \frac{\omega\left(\sum_{i=1}^{j} \ell_i\right)}{k} - \left[\frac{\omega 2\beta}{k} - \frac{2\beta}{k} + 2\beta\right] = \ell_{j+1},$$

$$\left[L + (\omega L - L)\left(1 + \frac{\omega}{k}\right)^{j}\right] - \left[\left(\frac{\omega 2\beta}{k} - \frac{2\beta}{k} + 2\beta\right)\left(1 + \frac{\omega}{k}\right)^{j}\right] = \ell_{j+1}.$$

and the claim follows by the definition of ℓ_{j+1}. □

D PROOFS OF LOWER BOUND RESULTS

This section formally proves the lower bound results for OPR-max, building on the proof for OPR-min provided in Section 5.2.

D.1 Proof of Theorem 9 (OPR-max Lower Bound)

PROOF OF THEOREM 9. Let ALG be a deterministic online algorithm for OPR-max, and suppose that the adversary uses the price sequence $u_1, \ldots, u_k$, which is exactly the sequence defined by (6). u_1 is presented to ALG, at most k times or until ALG accepts it. If ALG never accepts u_1, the remainder of the sequence is all L, and ALG achieves a competitive ratio of $\frac{ku_1 - 2\beta}{kL - 2\beta} = \omega$, as defined in (8).

If ALG accepts u_1, the next price presented is L, repeated at most k times *or until* ALG *switches to reject* L. After ALG has switched, u_2 is presented to ALG, at most k times or until ALG accepts it. Again, if ALG never accepts u_2, the remainder of the sequence is all L, and ALG achieves a competitive ratio of at least $\frac{ku_2 - 2\beta}{u_1 + (k-1)L - 4\beta} = \omega$, as defined in (8).

As the sequence continues, whenever ALG does not accept some u_i after it is presented k times, the adversary drops the price to L for the remainder of the sequence. Otherwise, if ALG accepts k prices before the end of the sequence, the adversary concludes by presenting U at least k times.

Observe that any ALG which does not immediately reject the first L presented to it after accepting some u_i obtains a competitive ratio strictly worse than ω. To illustrate this, suppose ALG has just accepted u_1, achieving a profit of $u_1 - \beta$ so far. The adversary begins to present L prices, and ALG accepts $y \leq (k-1)$ of these L prices before switching away. If $y = (k-1)$, ALG will accept k prices before the end of the sequence and achieve a competitive ratio of $\frac{kU - 2\beta}{u_1 + (k-1)L - 2\beta} > \omega$. Otherwise, if $y < (k-1)$, the profit achieved by ALG so far is at most $u_1 - 2\beta + yL$, while the profit achieved by ALG if it had immediately switched away ($y = 0$) would be $u_1 - 2\beta$ – since any price which might be accepted by ALG in the future should be $\geq L$, the latter case strictly improves the competitive ratio of ALG.

Assuming that ALG does immediately reject any L presented to it, and that ALG accepts some prices before the end of the sequence, the competitive ratio attained by ALG is at least $\frac{ku_{j+1} - 2\beta}{\sum_{i=1}^{j} u_i - (j+1)2\beta + (k-j)L} = \omega$, as defined in (8).

Similarly, if ALG accepts k prices before the end of the sequence, the competitive ratio attained by ALG is at least $\frac{kU - 2\beta}{\sum_{i=1}^{k} u_i - k2\beta} = \omega$, as defined in (8).

Since any arbitrary deterministic online algorithm ALG cannot achieve a competitive ratio better than ω playing against this adaptive adversary, our proposed algorithm DTPR-max is optimal.   □