# Movement Penalized Bayesian Optimization with Application to Wind Energy Systems

**Shyam Sundhar Ramesh**
ETH Zurich
shramesh@ethz.ch

**Pier Giuseppe Sessa**
ETH Zurich
sessap@ethz.ch

**Andreas Krause**
ETH Zurich
krausea@ethz.ch

**Ilija Bogunovic**
University College London
i.bogunovic@ucl.ac.uk

## Abstract

Contextual Bayesian optimization (CBO) is a powerful framework for sequential decision-making given side information, with important applications, e.g., in wind energy systems. In this setting, the learner receives context (e.g., weather conditions) at each round, and has to choose an action (e.g., turbine parameters). Standard algorithms assume no cost for switching their decisions at every round. However, in many practical applications, there is a cost associated with such changes, which should be minimized. We introduce the episodic CBO with movement costs problem and, based on the online learning approach for metrical task systems of Coester and Lee [19], propose a novel randomized mirror descent algorithm that makes use of Gaussian Process confidence bounds. We compare its performance with the offline optimal sequence for each episode and provide rigorous regret guarantees. We further demonstrate our approach on the important real-world application of altitude optimization for Airborne Wind Energy Systems. In the presence of substantial movement costs, our algorithm consistently outperforms standard CBO algorithms.

## 1 Introduction

Bayesian optimization (BO) is a well-established framework for sequential black-box function optimization that relies on Gaussian Process (GP) models [42] to sequentially learn and optimize the unknown objective. In many practical scenarios, however, one wants to additionally use available *contextual* information when making decisions. In this setting, at each round, the learner receives a context from the environment and has to choose an action based upon it. Previous works have developed contextual BO algorithms [32, 16, 31, 40], and applied them to various important applications, e.g., vaccine design, nuclear fusion, database tuning, crop recommender systems, etc.

A potential practical issue with these standard algorithms is that they assume no explicit costs for *switching* between their actions at every round. Frequent action changes can be extremely costly in many real-world applications. This work is motivated by the problem of real-time control of the altitude of an airborne wind energy (AWE) system.[1] In AWE systems, the wind speed is often only measurable at the system's altitude, and determining the optimal operating altitude of an AWE system as the wind speed varies represents a challenging problem. Another fundamental challenge is that additional energy is required for adjusting the altitude, which makes the frequent altitude changes costly. Consequently, this work is motivated by the following question: *How can we efficiently learn*

[1] AWE system is a wind turbine with a rotor supported in the air without a tower that can benefit from the persistence of wind at different high altitudes [22].

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

*to optimize the AWE system's operating altitude despite varying wind conditions while minimizing the energy cost associated with turbine altitude changes?*

In this work, we formalize the *movement penalized* contextual BO problem. When the switching cost is a *metric* (distance function), we propose a novel algorithm that effectively combines ideas from BO with the online learning strategies proposed in [19] for solving the so-called *metrical task system* (MTS) problem [11]. Furthermore, our algorithm relies solely on noisy point evaluations (i.e., bandit feedback), allows for arbitrary context sequences, and besides the standard exploration-exploitation trade-off, it also balances the movement costs. As a result, it outperforms the standard movement-cost-agnostic contextual BO algorithms as well as movement-conservative baselines (see Fig. 1).

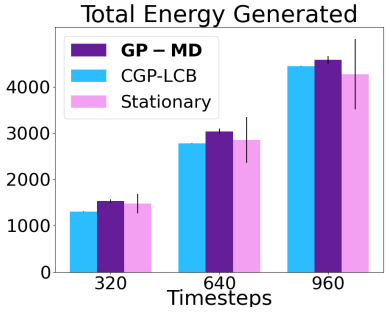



Figure 1: Total energy generated by the AWE system when operated with GP-MD (proposed in this work), CGP-LCB [32] and a stationary baseline. The stationary baseline employs no-learning and does not incur movement costs. GP-MD and CGP-LCB learn an operating strategy, but GP-MD outperforms CGP-LCB since it also considers movement costs.

**Related Work.** Bayesian optimization (BO) refers to a sequential approach for optimizing an unknown objective (cost function) from noisy point evaluations. A great number of BO methods have been developed over the years (e.g., [38, 50, 17]). While the focus of standard BO approaches is mainly on optimizing the unknown objective cost function, in this work, we additionally focus on penalizing frequent action changes. This problem makes the most sense in the *contextual* BO setting [32], where the main objective changes with the observed contextual information and the learner also seeks to minimize the cost associated with frequent changes of its actions. Several works have tried to incorporate such and similar cost functions in the BO setting. [35] explicitly consider switching costs based on how deep into the pipeline the change in variable occurs. But they do not consider the contextual setting that is essential for our wind energy application and hence our work is not directly comparable with theirs. Moreover, it assumes that the system of interest has a modular structure (as detailed in [35, Section-3]). In such a modular setting, the cost at each round is the number of modules that has to be changed rather than the amount of change of each variable. Other works include cost-aware and multi-objective sampling strategies in various settings such as batch [29, 27], multi-fidelity [10, 28, 49], multi-objective optimization [2] and dynamic programming [34, 33]. Finally, two recent works [24, 15] consider the problem of switching cost minimization in Bayesian optimization. They both consider the non-contextual setting, and while [24] lacks theoretical guarantees, [15] does not explicitly consider the cost in the regret definition. Similarly, we consider movement/switching costs, but unlike these previous work, we specifically focus on minimizing the movement costs in the contextual setting.

The *Metrical Task Systems* (MTS) problem [11] is a sequential decision-making problem widely studied in the *online learning* literature. At each round, the learner observes a service cost function, chooses an action, and incurs the corresponding cost together with a movement cost penalizing the distance (according to some metric) between the current action and the one chosen at the previous round. The MTS problem is directly related to our problem setting (see Section 2.2). After a long series of works, in [14] and [19], an $\mathcal{O}\big((\log n)^2\big)$-competitive algorithm for MTS on any finite metric space was established. The approach of Coester and Lee [19] relies on a tree representation of the decision space and an action randomization scheme via a mirror descent procedure. In contrast to our setting, these works assume that the service costs are *known* (i.e., they assume the *full-information* feedback). Moreover, in the online optimization literature, other related works study the online *convex* optimization with switching costs [26, 48, 25, 36] and convex body chasing problems [12, 5, 43, 13]. We make no use of convexity and take a model-based (GP) approach to learn about the unknown service costs. [37] consider non-convex objective functions, however, they impose certain conditions on the objective that are not easily modeled via GP as done in our work.

The use of GP confidence bounds in online learning settings has been previously explored, e.g., in repeated multi-agent [44, 47] and sequential games [46], and to discover randomized max-min strategies [45]. However, none of these works has considered movement costs in the objective. This makes our problem significantly different from the aforementioned ones, and requires a suitable action randomization scheme that can trade-off exploration, exploitation, and movement costs simultaneously.

Various works have applied Bayesian Optimization in the context of *wind energy systems* before. [41, 21, 4] use BO techniques to maximize the total energy yield in a wind farm in a cooperative or a closed-loop framework. [39] use it to tune the parameters of the wind turbine to learn the effective wind speed. [6] use BO for plant design (i.e., physical system design) of airborne wind energy systems. [51] apply BO and other regression techniques to predict short-term wind energy production to make informed production offers. Finally, [3] survey several different methods for efficient wind-power prediction using machine-learning methods including BO. Our problem formulation differs from the above mentioned works, since we explicitly consider movement energy loss caused by the altitude change of the system.

**Contributions.** We formally introduce Bayesian optimization with movement costs and propose a novel GP-MD algorithm (in Algorithm 1). GP-MD combines the online mirror descent (MD) algorithm with shrinking Gaussian Process (GP) confidence bound to decide on which point to evaluate next. In our theoretical analysis, we establish rigorous sublinear regret guarantees for our algorithm by combining techniques from Bayesian optimization and metrical task systems approaches [19]. Finally, we demonstrate that GP-MD is able to successfully outperform previous contextual Bayesian optimization approaches on both synthetic and real-world data in the presence of movement costs. In particular, we consider the application to airborne wind energy systems and demonstrate that GP-MD can effectively operate such a system by considering movement costs and varying environmental conditions.

## 2 Problem Statement

Let $f : \mathcal{X} \times \mathcal{E} \to \mathbb{R}_+$ be an *unknown* cost function defined over $\mathcal{X} \times \mathcal{E} \subset \mathbb{R}^p$, where $\mathcal{X}$ is a finite set of actions, i.e., $|\mathcal{X}| = n$, and $\mathcal{E}$ represents convex and compact space of contexts. We denote the *known* metric (i.e., distance function) of $\mathcal{X}$ as $d(\cdot, \cdot)$, and similarly to other works in Bayesian optimization (e.g., [50, 17]) assume that the target cost function $f$ belongs to a reproducing kernel Hilbert space (RKHS) $\mathcal{H}_k$ of functions (defined on $\mathcal{X} \times \mathcal{E}$), that corresponds to a known kernel $k : (\mathcal{X} \times \mathcal{E}) \times (\mathcal{X} \times \mathcal{E}) \to \mathbb{R}_+$ with $k((x, e), (x', e')) \leq 1$ for any action-context pair. In particular, we assume that for some known $B > 0$, the target cost $f$ has a bounded RKHS norm, i.e., $f \in \mathcal{F}_k = \{f \in \mathcal{H}_k : \|f\|_k \leq B\}$. Also, we assume that the diameter of $\mathcal{X}$ ($\max_{x,x' \in \mathcal{X}} d(x, x')$) is bounded and denote it by $\psi$.

We consider an episodic setting, wherein each episode runs over a finite time horizon $H$. Let the *initial state* of the system in the first episode correspond to action $x_{0,1} \in \mathcal{X}$. At the end of every episode, the system resets to a new given initial action $x_{0,m} \in \mathcal{X}$ where $m \in \{1, 2, \ldots, N_{ep}\}$ denotes the episode index. In each episode $m$ and at every time step $h \in \{1, 2, \ldots, H\}$, the environment reveals the context $e_{h,m} \in \mathcal{E}$ to the learner. We make no assumptions on the context sequence provided by the environment (i.e., it can be arbitrary and different across episodes). The learner then chooses $x_{h,m} \in \mathcal{X}$ and observes the noisy function value:

$$y_{h,m} = f(x_{h,m}, e_{h,m}) + \xi_{h,m}, \tag{1}$$

where $\xi_{h,m} \sim \mathcal{N}(0, \sigma^2)$ with known $\sigma$, and independence over time steps. The goal of the learner is to minimize the cost incurred over the rounds in every episode, but at the same time to minimize the distance between its subsequent decisions as measured by $d(x_{h-1,m}, x_{h,m})$.

Let $D_m = \{x_{1,m}, x_{2,m}, \ldots, x_{H,m}\}$ denote the set of actions chosen by the learner over $H$ rounds in episode $m$. We recall that each action $x_{h,m} \in D_m$ is chosen after observing the corresponding context $e_{h,m}$. The objective is to minimize the cumulative episodic cost for each episode $m$,

$$\mathrm{cost}_m(D_m) = \underbrace{\sum_{h=1}^{H} f(x_{h,m}, e_{h,m})}_{S_m(D_m)} + \underbrace{\sum_{h=1}^{H} d(x_{h,m}, x_{h-1,m})}_{M_m(D_m)}, \tag{2}$$

where we refer to the two terms in Eq. (2) as *service cost* $S_m$ and *movement cost* $M_m$.

When $f$ is known, the problem can be seen as a MTS instance as detailed in Section 2.2. Even in such a case, we cannot hope to solve this problem optimally, and nearly-optimal approximate algorithms were recently proposed (see Coester and Lee [19]). Hence, the learner's performance in episode $m$ is measured via $(\alpha, \beta)$-approximate regret:

$$r_m^{\alpha,\beta} = \mathrm{cost}_m(D_m) - \alpha \cdot \mathrm{cost}_m(D_m^*) - \beta, \tag{3}$$

where $D_m^* := \arg\min_{D \subset \mathcal{X},\, |D|=H} \text{cost}_m(D)$ is the offline optimal action sequence obtained assuming the knowledge of the true sequence of contexts $\{e_{h,m}\}_{h=1}^H$ *in advance*, and $\alpha$ and $\beta$ are approximation constants (independent of $N_{ep}$). In contrast, in our setting, the learner only gets to see the *current context* when making a decision and has no knowledge about the future ones.

After $N_{ep}$ episodes, the total cumulative regret is defined as

$$R_{N_{ep}}^{\alpha,\beta} = \sum_{m=1}^{N_{ep}} r_m^{\alpha,\beta}. \tag{4}$$

We seek an algorithm whose total cumulative regret grows sublinearly in $N_{ep}$, so that $\lim_{N_{ep}\to\infty} R_{H,N_{ep}}^{\alpha,\beta}/N_{ep} = 0$, for any set of initial states $\{x_{0,m}\}_{m=1}^{N_{ep}} \subset \mathcal{X}$.

### 2.1 Gaussian Process Model

In standard Bayesian optimization, a surrogate Gaussian Process model is typically used to model the target cost function. A Gaussian Process $GP(\mu(\cdot), k(\cdot,\cdot))$ over the input domain $\mathcal{X}\times\mathcal{E}$, is a collection of random variables $(f(x,e))_{x\in\mathcal{X},e\in\mathcal{E}}$ where every finite number of them $(f(x_i,e_i))_{i=1}^n$, $n \in \mathbb{N}$, is jointly Gaussian with mean $\mathbb{E}[f(x_i)] = \mu(x_i,e_i)$ and covariance $\mathbb{E}[(f(x_i,e_i)-\mu(x_i,e_i))(f(x_j,e_j)-\mu(x_j,e_j))] = k((x_i,e_i),(x_j,e_j))$ for every $1 \le i,j \le n$.

BO algorithms typically use zero-mean GP priors to model uncertainty in $f$, i.e., $f \sim GP(0, k(\cdot,\cdot))$, and Gaussian likelihood models for the observed data. As more data points are observed, GP (Bayesian) posterior updates are performed in which noise variables are assumed to be drawn independently across $t$ from $\mathcal{N}(0,\lambda)$. Here, $\lambda$ is a hyperparameter that might be different from the true noise variance $\sigma^2$. More precisely, given a sequence of previously queried points and their noisy observations the posterior is again Gaussian, with the posterior mean and variance given by:

$$\mu_t(x,e) = k_t(x,e)^T(K_t + \lambda I_t)^{-1} Y_t \tag{5}$$

$$\sigma_t^2(x,e) = k((x,e),(x,e)) - k_t(x,e)^T(K_t + \lambda I_t)^{-1} k_t(x,e), \tag{6}$$

where $Y_t := [y_1,\ldots,y_t]$ denotes a vector of observations, $K_t = [k((x_s,e_s),(x_{s'},e_{s'})]_{s,s'\le t}$ is the corresponding kernel matrix, and $k_t(x,e) = [k((x_1,e_1),(x,e)),\ldots,k((x_t,e_t),(x,e))]^T \in \mathbb{R}^{t\times 1}$.

**Maximum Information Gain.** In the standard Bayesian optimization, the main quantity that characterizes the complexity of optimizing the target cost function is the maximum information gain [50] defined at time $t$ as:

$$\gamma_t = \max_{\{(x_i,e_i)\}_{i=1}^t} I(Y_t; f), \tag{7}$$

where $I(Y_t; f)$ denotes the mutual information between random observations $Y_t$ and GP model $f$. The mutual information for the GP model is given as:

$$I(Y_t; f) = \frac{1}{2}\log\det(I_t + \lambda^{-1}K_t). \tag{8}$$

This quantity is kernel-specific and for compact and convex domains $\gamma_t$ is sublinear in $t$ for various classes of kernel functions [50] as well as for kernel compositions (e.g., additive kernels in [32]).

**Confidence Bounds.** We also use the following result ([50, 1, 17]) that is frequently used in Bayesian optimization to provide confidence bounds around the unknown function.

**Lemma 1.** *Assume the $\sigma$-sub-Gaussian noise model as in Eq.* (1)*, and let $f$ belong to $\mathcal{F}_k$. Then, the following holds with probability at least $1-\delta$ simultaneously over all $t \ge 1$ and $x \in \mathcal{X}$, $e \in \mathcal{E}$:*

$$|\mu_t(x,e) - f(x,e)| \le \beta_t \sigma_t(x,e), \tag{9}$$

*where $\beta_t = \frac{\sigma}{\lambda^{1/2}}\sqrt{2\ln(1/\delta) + 2\gamma_t} + B$, and $\mu_t$ and $\sigma_t$ are defined in Eqs.* (5) *and* (6) *with $\lambda > 0$.*

Based on the previous, we also define the lower confidence bound for every $x \in \mathcal{X}, e \in \mathcal{E}$ as:

$$\text{lcb}_t(x,e) := \mu_t(x,e) - \beta_t\sigma_t(x,e). \tag{10}$$

We use $\text{lcb}_m(x,e)$ when it is computed based on data collected before episode $m$.

**Algorithm 1** GP-MD

**Require:** Action space $\mathcal{X}$, kernel function $k(\cdot,\cdot)$, metric $d(\cdot,\cdot)$
Run FRT$(\mathcal{X}, d(\cdot,\cdot))$ and obtain $\tau$-HST $\mathcal{T} = (V, E)$ with leaves $\mathcal{L} = \mathcal{X}$
**for** $m = 1, \dots, N_{ep}$ **do**
  Receive $x_{0,m}$ and initialize $z_{0,m}$ (Eq. (29)), conditional prob. $q_0 = \Delta^{-1}(z_{0,m})$ as in Eq. (15)
  **for** $h = 1, \dots, H$ **do**
    Observe context $e_{h,m}$ and initialize costs: $\text{lcb}_m(v, e_{h,m}) = 0, \forall v \in V \backslash \mathcal{L}$
    **for** $u \in \mathcal{OD}(V \backslash \mathcal{L})$ **do**
      Update vertex prob. $q_h^{(u)}$ from $q_{h-1}^{(u)}$ and $\text{lcb}_m(\cdot, e_{h,m})$ via Mirror Descent (Eq. (13))
      Update cost for vertex $u$:

$$\text{lcb}_m(u, e_{h,m}) = \langle q_h^{(u)}, \text{lcb}_m(\cdot, e_{h,m}) \rangle = \sum_{\nu \in \mathcal{C}(u)} q_{h,\nu} \cdot \text{lcb}_m(\nu, e_{h,m})$$

    **end for**
    Compute prob. vector $z_{h,m} = \Delta(q_h)$ (Eq. (15)) and leaves' prob. $l(z_{h,m})$ (Eq. (11))
    Estimate optimal coupling $\zeta_{h-1,h,m}$ between $l(z_{h-1,m})$ and $l(z_{h,m})$ as in Eq. (12)
    Sample action $x_{h,m} \sim \zeta_{h-1,h,m}(\cdot | x_{h-1,m})$ and observe $y_{h,m} = f(x_{h,m}, e_{h,m}) + \xi_{h,m}$
  **end for**
  Update $\mu_{m+1}(\cdot,\cdot)$ and $\sigma_{m+1}(\cdot,\cdot)$ as per Eq. (5) and Eq. (6)
**end for**

### 2.2 Relation to Metrical Task Systems (MTS)

When $f$ is known, our optimization objective in Eq. (2) can be seen as a particular type of MTS problem, where $f(\cdot, e_{h,m})$ is the MTS service cost that changes for every $h$ and $m$. Compared to a standard MTS (see Appendix A), our problem formulation is more challenging since the learner can only learn about $f$ from previously observed data. The approach proposed in this paper builds on the algorithm by Coester and Lee [19] for standard MTS problems. However, to cope with the aforementioned challenge, our approach exploits the regularity assumptions regarding $f$ and utilizes the constructed lower confidence bounds Eq. (10) to *hallucinate* information about the unavailable service cost at each round. Before presenting our overall approach, we describe a preliminary step proposed by [19], which consists of representing our metric space $(\mathcal{X}, d)$ by a ***Hierarchically Separated Tree*** (HST) metric space.

**HST metric space.** Consider a tree $\mathcal{T} = (V, E)$ with root $r$, leaves $\mathcal{L} \subset V$ and non-negative weights $w_v$, for each $v \in V$, which are non-increasing along root-leaf paths. Let $d_{\mathcal{T}}(l, l')$ denote a distance metric between any two leaves $l, l' \in \mathcal{L}$ given as the sum of the encountered weights on the path from $l$ to $l'$ (see Fig. 4). $(\mathcal{L}, d_{\mathcal{T}})$ is a HST metric space, and $\tau$-HST metric space if the weights are exponentially decreasing, i.e., $w_u \leq w_v/\tau$, with $v$ being the parent of $u$.

Similarly to [19], we use the algorithm from [23] (which we name via the authors' surnames as FRT) to approximate the given metric space $(\mathcal{X}, d)$ by a $\tau$-HST one. In particular, we use FRT in Algorithm 1 as a computationally efficient preprocessing step to create a tree $\mathcal{T}$ with leaves $\mathcal{L}$ corresponding to actions in $\mathcal{X}$, distance metric $d_{\mathcal{T}}$, and root node $r$. We explain the intrinsic MTS motivation for this preprocessing step in Appendix B.1, and defer additional details to Appendix B.2.

## 3 The GP-MD Algorithm

In this section, we introduce GP-MD, a novel algorithm for the contextual BO problem with movement costs defined in Section 2. At each episode $m$ and round $h$, the state of GP-MD can be summarized by a vector of probabilities $z_{h,m} \in K_{\mathcal{T}}$ over the vertices of $\mathcal{T}$, where $K_{\mathcal{T}} := \Big\{ z \in \mathbb{R}_+^{|V|} : z_r = 1, z_u = \sum_{\nu \in \mathcal{C}(u)} z_\nu \quad \forall u \in V \backslash \mathcal{L} \Big\}$, and $\mathcal{C}(u)$ denotes the children of $u$. Each entry $z_\nu$ represents the probability that the selected action $x_{h,m}$ belongs to the leaves of the subtree rooted at $\nu$, i.e., $z_\nu = \mathbb{P}(x_{h,m} \in \mathcal{L}(\nu))$. Below, we specify how $z_{h,m}$ is computed at each round. Moreover,

given any $z \in K_{\mathcal{T}}$, the vector

$$l(z) := [z_l,\ l \in \mathcal{L}] \in [0,1]^n, \tag{11}$$

defines a probability distribution over the leaves $\mathcal{L}$, and hence the actions $\mathcal{X}$. As in [19], given probability vectors $z_{h,m}$ and $z_{h-1,m}$, GP-MD computes the *minimal distance* distribution

$$\zeta_{h-1,h,m} = \underset{\zeta \in \Pi(l(z_{h-1,m}), l(z_{h,m}))}{\arg\inf} \mathbb{E}_{\zeta}[d_{\mathcal{T}}(U_{h-1,m}, U_{h,m})], \tag{12}$$

where $U_{h-1,m}$ and $U_h$ are random variables having marginals $l(z_{h-1,m})$ and $l(z_{h,m})$ respectively. Finally, action $x_{h,m}$ is sampled from the conditional minimal distance distribution $x_{h,m} \sim \zeta_{h-1,h,m}(\cdot|x_{h-1,m})$ (Line 13 in Algorithm 1). At the end of each episode $m$, the newly observed data are then used to update posterior mean and standard deviation about the cost function.

Finally, we describe how probability vectors $z_{h,m}$ are computed at each round, a key step of GP-MD (Lines 8–12 in Algorithm 1). We follow the recursive Mirror Descent (MD) procedure proposed by [19], with the important difference that we are dealing with an *unknown* context-dependent cost function. Hence, we make use of the Gaussian process model and corresponding confidence estimates as defined in Section 2.1.

To obtain probabilities $z_{h,m}$, we consider *conditional* probability vectors $q \in Q_{\mathcal{T}}$, where $Q_{\mathcal{T}}$ is the set of valid conditional probabilities $Q_{\mathcal{T}} := \Big\{ q \in \mathcal{R}_+^{|V \setminus r|} : \sum_{\nu \in \mathcal{C}(u)} q_\nu = 1 \quad \forall u \in V \setminus \mathcal{L} \Big\}$. For each vertex $\nu$ with parent $u$, $q_\nu$ represents the conditional probability $\mathbb{P}(x_{h,m} \in \mathcal{L}(\nu) | x_{h,m} \in \mathcal{L}(u))$. Moreover, given $q_h \in Q_{\mathcal{T}}$ we define the vector $q_h^{(u)} := [q_{h,\nu},\ \nu \in \mathcal{C}(u)]$ as the conditional distribution over children of $u$, and let $Q_{\mathcal{T}}^{(u)}$ be the set of all valid distributions $q_h^{(u)}$.

In each episode $m$, conditional probability vector $q_h$ for round $h$ is obtained recursively, from leaves to root, as a function of $q_{h-1}$, the observed context $e_{h,m}$, and the current estimate about the cost associated to each particular vertex. More precisely, let $\mathcal{OD}(V \setminus \mathcal{L})$ be a topological ordering of the internal vertices $V \setminus \mathcal{L}$ so that every child in $\mathcal{T}$ occurs before its parent. Then, for each $u \in \mathcal{OD}(V \setminus \mathcal{L})$ conditional probabilities $q_h^{(u)}$ are obtained via the Mirror Descent update:

$$q_h^{(u)} = \underset{p \in Q_{\mathcal{T}}^{(u)}}{\arg\min} \Big\{ D^{(u)}(p \| q_{h-1}^{(u)}) + \langle p, \mathrm{lcb}_m^{(u)}(\cdot, e_{h,m}) \rangle \Big\}. \tag{13}$$

Function $D^{(u)}$ is the Bregman divergence with respect to a suitable potential function (see Appendix D.1), while $\mathrm{lcb}_m^{(u)}(\cdot, e_{h,m}) := [\mathrm{lcb}_m(\nu, e_{h,m}),\ \forall \nu \in \mathcal{C}(u)]$ is a lower confidence bound estimate of the costs corresponding to children of vertex $u$. For $v \in \mathcal{L}$, $\mathrm{lcb}_m(\nu, e_{h,m})$ are obtained by the GP-regression techniques outlined in Section 2.1, while for internal vertices these are computed recursively from their children nodes as:

$$\mathrm{lcb}_m(u, e_{h,m}) := \sum_{\nu \in \mathcal{C}(u)} q_{h,\nu} \mathrm{lcb}_m(\nu, e_{h,m})\,. \tag{14}$$

The movement cost is primarily controlled by this usage of Bregman divergence based mirror descent. Also, sampling from the conditional minimal distance distribution further restricts movement between alternate actions. Once the vector of conditional probabilities $q_h \in Q_{\mathcal{T}}$ has been updated, we can obtain the corresponding probability vector $z_{h,m}$ via the mapping $\Delta : Q_{\mathcal{T}} \to K_{\mathcal{T}}$ such that:

$$z = \Delta(q) \ \ \Rightarrow \ \ z_\nu = z_u q_\nu \quad \forall u \in V \setminus \mathcal{L},\ \nu \in \mathcal{C}(u). \tag{15}$$

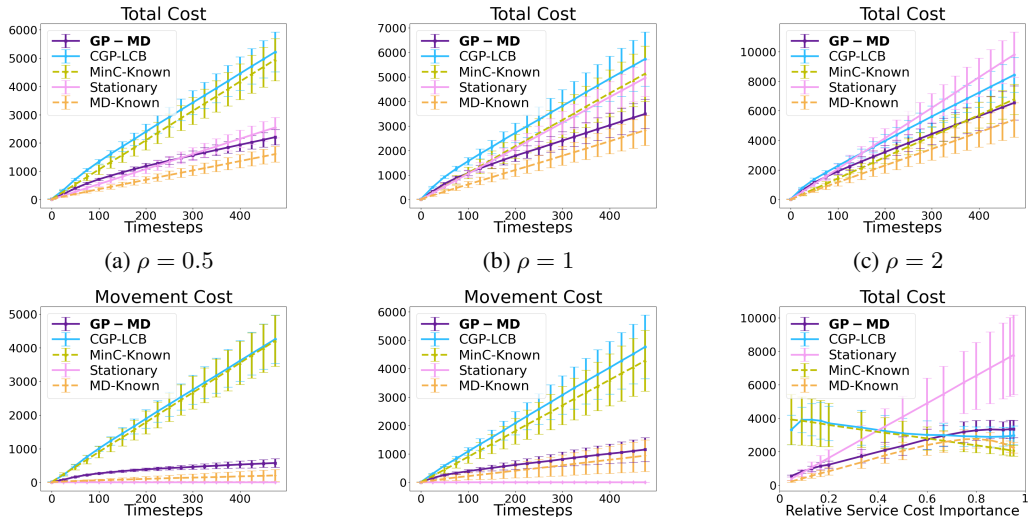
### 3.1 Theoretical Guarantees

Our main theorem bounds the cumulative regret of GP-MD.

**Theorem 1.** *Let $\mathcal{X}$ be represented by a $\tau$-HST space with $\tau > 4$ (Line 2 of Algorithm 1), and set $\delta \in (0,1)$. Then, with probability at least $1-\delta$, the regret of* GP-MD *over $N_{ep}$ episodes is bounded by*

$$R_{N_{ep}}^{\alpha,\beta} = \mathcal{O}\Big( \beta_{N_{ep}} \big( N_{ep} H^2 \gamma_{HN_{ep}} + H \log(\tfrac{H}{\delta}) \big)^{\frac{1}{2}} + H(B+\psi) \log\big( \tfrac{N_{ep} \log(N_{ep})}{\delta} \big) \Big),$$

*with approximation factors $\alpha = \mathcal{O}\big((\log n)^2\big)$ and $\beta = \mathcal{O}(1)$. Here, $H$ is the episodes' length, $\beta_{N_{ep}}$ is the confidence level from Lemma 1, and $\gamma_{HN_{ep}}$ is the maximum information gain defined in Eq. (7).*




(a) $\rho = 0.5$ (b) $\rho = 1$ (c) $\rho = 2$

(d) $\rho = 0.5$ (e) $\rho = 2$ (f) Varying $\rho$

Figure 2: Total and movement cost performance of algorithms on synthetic functions for varying importance of movement/service cost (i.e., different $\rho$ values). GP-MD outperforms CGP-LCB in terms of total incurred cost, and its performance closely follows one of the idealized benchmark MD-KNOWN. GP-MD also minimizes the movement cost while CGP-LCB suffers from significant movements (Figs. 2d and 2e). The performance of GP-MD remains robust when the movement cost importance in the total cost objective diminishes (Fig. 2f).

For most of the popularly used kernels, Thm. 1 can be made more explicit by substituting bounds on $\gamma_{HN_{ep}}$ (e.g., in the case of linear kernel and compact domain, we have $\gamma_t = O(p \log t)$, while for squared-exponential kernel it holds $\gamma_t = O((\log t)^{p+1})$ [50]; see also Section 2.1). In such cases, we make the following two important observations regarding our result: i) The obtained regret bound is sublinear in the number of episodes $N_{ep}$ and hence $\lim_{N_{ep}\to\infty} R^{\alpha,\beta}_{N_{ep}}/N_{ep} = 0$; ii) The bound is independent of the input space size, i.e., the number of actions $n$ (although the approximation factor $\alpha$ depends logarithmically on $n$, similarly to [19]). These imply that GP-MD approaches $\alpha$-competitive ratio performance of the MTS algorithm by [19], while learning about the service cost from noisy point evaluations (i.e., bandit feedback) only. Finally, in our analysis, we treat $H$ as constant.

Proof of Thm. 1 is detailed in Appendix E. Next, we outline some main steps. We make use of the competitive ratio guarantees for the used Mirror Descent algorithm from [19, Corollary 4] to bound the expected hallucinated service cost $\sum_{h=1}^{H} \langle \text{lcb}_m(\cdot, e_{h,m}), l(z_{h,m}) \rangle$. Here, we use the fact that $\text{lcb}_m(\cdot,\cdot)$ does not change within an episode. Since this is not the actual service cost, we bound $\sum_{h=1}^{H} \langle f(\cdot, e_{h,m}), l(z_{h,m}) \rangle$ by $\sum_{h=1}^{H} \langle \text{lcb}_m(\cdot, e_{h,m}), l(z_{h,m}) \rangle$ with an additional learning error. The sum of the learning errors over all episodes can be rewritten as $\sum_{m=1}^{N_{ep}} \sum_{h=1}^{H} \langle \sigma^2_m(\cdot, e_{h,m}), l(z_{h,m}) \rangle$. We use the concentration of the conditional mean result from [30, Lemma 3] to upper bound it by the actual realizations $\sum_{m=1}^{N_{ep}} \sum_{h=1}^{H} \sigma^2_m(x_{h,m}, e_{h,m})$, and use the result of [18, Lemma-2], to further upper bound it with the maximum information gain quantity $\gamma_{N_{ep}H}$.

Finally, the movement cost can also be bounded similar to that of the previously mentioned expected hallucinated service cost $\sum_{h=1}^{H} \langle \text{lcb}_m(\cdot, e_{h,m}), l(z_{h,m}) \rangle$ with an extra $\alpha = \mathcal{O}\big((\log n)^2\big)$ factor using [19, Corollary 4].

## 4 Experiments

This section provides numerical results on synthetic and real-world data. We compare the performance of our GP-MD algorithm with the following baselines:

- STATIONARY selects the stationary strategy $x_h = x_0$ for all $h$,
- CGP-LCB [32] neglects the movement cost and sets $x_h = \arg\min_x \text{lcb}_h(x, e_h)$ for all $h$,
- MINC-KNOWN assumes $f(\cdot)$ is known and chooses $x_h = \arg\min_x f(x, e_h)$, and
- MD-KNOWN assumes $f(\cdot)$ is known and runs mirror descent from [19] on $f(\cdot, e_h)$.

MD-KNOWN and MINC-KNOWN unrealistically assume that $f(\cdot)$ is known and can be seen as upper-bound for the achievable performance of GP-MD and CGP-LCB, respectively. We use the same constant value $\beta = 2.0$ for the exploration parameter in both GP-MD and CGP-LCB (since the theoretical worst-case bounds are found to be overly pessimistic and can impede performance [50]). We run the algorithms over a single episode (as done in CGP-LCB). We also discover that updating the confidence bounds after every timestep in GP-MD leads to better performance in practice.

**Synthetic experiments.** We consider synthetic experiments, where the objective function is a random GP sample. The considered action space $\mathcal{X}$ is a subset of $[0, 1]^2$ consisting of $400$ points that form the uniform grid, while the context space $\mathcal{E}$ consists of $40$ contexts that are uniformly sampled from $(0, 1)$. We sample objective function (i.e., actual cost) $f : \mathcal{X} \times \mathcal{E} \to \mathbb{R}$ from a $GP(0, k)$, where $k$ is a squared exponential kernel with lengthscale parameter set to $l = 0.2$. We use the Euclidean distance between the domain points as the movement cost, calculate the distance matrix (between every pair of points) and the average movement cost. We subtract the minimum value from $f$ and scale it such that the average function value is equal to the average movement cost. We also set the noise parameter to $1\%$ of the function range. We introduce the trade-off parameter $\rho \in \{0.25, 0.5, 1, 2, 4\}$ that only multiplies the service cost, i.e, $\rho f(x, e)$, but not the movement cost. This is to test the performance of the algorithms for varying importance of the service/movement costs. For each $\rho$ we sample $25$ different functions and run the algorithms for $500$ timesteps wherein at each step the contexts are randomly sampled.

In Fig. 2a-Fig. 2e, we show the total cumulative cost as a function of timesteps for different $\rho$ values. Then, in Fig. 2f, we show the performance of the algorithms (for known kernel parameters) when run for $800$ timesteps for varying importance of the service and movement costs. In particular, we consider a convex combination of the service and movement costs, where we set the respective coefficients multiplying these two objectives as $\rho/(1 + \rho)$ and $1/(1 + \rho)$.

As shown in Fig. 2a-Fig. 2c, the performance of GP-MD is generally close to the one of the idealized, unrealistic benchmark MD-KNOWN, which, as expected, performs the best. The stationary baseline performs comparably when $\rho$ is small, while its performance deteriorates for larger values. As expected, both MINC-KNOWN and CGP-LCB incur higher total costs than GP-MD when the movement cost is of the higher or same relative importance as the service cost (i.e., $\rho \in \{0.5, 1.0\}$), while the performance gap slowly decreases when the service cost becomes dominant ($\rho = 2.0$). In Figs. 2d and 2e, we also show the corresponding movement costs, and observe that movement cost ignorant CGP-LCB incurs significant movement costs, while our GP-MD successfully minimizes the movement costs. Finally, in Fig. 2f, we observe that the performance of GP-MD is robust, i.e., it clearly outperforms CGP-LCB whenever the movement cost dominates the total cost objective, while its performance remains comparable to the one of CGP-LCB (that is built to minimize service cost) when the movement cost becomes dominated by the service cost.

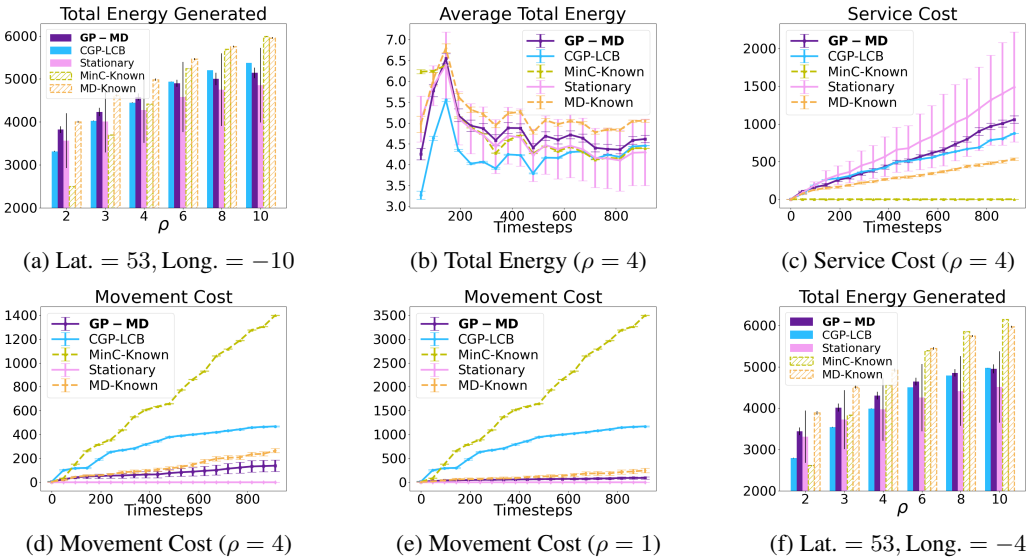
### 4.1 Altitude Optimization in AWE Systems

In airborne wind energy (AWE) systems, the turbine's operating altitude can be changed depending on the wind pattern. We follow the setup of Baheri et al. [7] that applied CGP-LCB [32] which ignores movement-costs, to learn this control task. In this section, we use a dataset from [9] which contains wind-speed information over various locations in Europe for a period ranging from 2011 to 2017, and also includes measurements at different altitudes per location. We consider the wind speed data from the second half of 2016. Our goal is to maximize the generated energy, while taking into account the energy loss due to moving the turbine from one altitude to another.

We consider $25$ different altitudes (ranging from 10m to 1600m) as the action space and the context space to be hours in the day (i.e., $24$ values). We define our unknown service objective function to be $f(x, t) = \max_{x'}(E_S(x', t)) - E_S(x, t)$ where $E_S(x, t)$ denotes the energy generated based on the windspeed at altitude $x$ and time $t$. Based on the discrete-time power generation formula from Baheri et al. [7] (Eq. (10)), we have

$$E_S(x, t) = \big(c_1(\min\{V_w(x, t), V_r\})^3 - c_2 V_w^2(x, t)\big)\Delta t. \tag{16}$$

Here $V_w(x, t)$ denotes the windspeed at altitude $x$ and time $t$, and $V_r$ denotes the rated windspeed of the turbine. The constants $c_1 = 0.0579$ and $c_2 = 0.09$ are system dependent. This corresponds to the energy generated at a particular altitude $x$ for $\Delta t$ time. Similarly to Baheri et al. [7], we use $\Delta t = 60$ since we consider intervals of one hour length. Next, we define the movement cost to be




(a) Lat. $= 53$, Long. $= -10$ (b) Total Energy ($\rho = 4$) (c) Service Cost ($\rho = 4$)

(d) Movement Cost ($\rho = 4$) (e) Movement Cost ($\rho = 1$) (f) Lat. $= 53$, Long. $= -4$

Figure 3: AWE altitude optimization task; Fig. 3a: Total energy generated for 960 hours based on the wind data at a single location (Latitude$= 53$ and Longitude$= -10$). GP-MD outperforms previously used CGP-LCB (that optimizes for service costs only) for a range of $\rho$ values that favor the service against the movement cost. Fig. 3b: The average total generated energy over 960 hours. Figs. 3c and 3d: The service and movement costs for $\rho = 4$. Fig. 3e: The movement costs for $\rho = 1$. The movement energy loss is slightly lower for GP-MD as compared to $\rho = 4$ due to higher importance towards movement cost reduction. Fig. 3f: Same as Fig. 3a, albeit by using wind data from a different location (Latitude$= 53$, Longitude$= -4$).

the energy lost in changing altitude (from $x$ to $x'$):

$$E_M(x, x') = c_3 V_r^2 |x - x'|, \tag{17}$$

where $c_3 = 0.15$ (see Appendix F for more details).[2]

We assume that a wind speed gauge is attached to the turbine, and the operator knows the wind speed at the current altitude. Hence, instead of directly learning $f(x, t)$, we learn $V_w(x, t)$ and use its confidence bounds to calculate the confidence bounds of $f(x, t)$. To learn $V_w(x, t)$, we normalize the inputs, and fit a GP with RBF kernel (lengthscale=3.67, outputscale=6.85 and noise parameter=2.73).

We run the algorithms for different $\rho$ for 960 timesteps, where again $\rho$ is used to multiply $E_S$. For each $\rho$, the algorithms were initiated with every possible starting point (25 different altitudes), and ran for 3 iterations. Based on this we plot the total energy generated w.r.t. varying $\rho$ in Fig. 3. In Figs. 3a and 3f, we show the performance of the algorithms at two different locations (we also consider additional locations and time periods in Appendix F). We use different values of $\rho > 1$ to show the robustness of our algorithm (as $\rho$ increases, the importance of the service cost w.r.t. the movement cost in the overall objective increases). Our algorithm outperforms CGP-LCB for a range of $\rho$ values. As $\rho$ keeps increasing, we observe that MINC-KNOWN closes the performance gap to MD-KNOWN, and the same is happening with CGP-LCB w.r.t. GP-MD. In Fig. 3b, we focus on a particular $\rho = 4$, and notice that GP-MD performs better than CGP-LCB and STATIONARY algorithm at this location. In Fig. 3c, we plot the service cost and observe that both learning algorithms GP-MD and CGP-LCB have lower service cost than the STATIONARY baseline. We also note that due to the implicit service cost definition, the MINC-KNOWN baseline achieves zero service cost. In Figs. 3d and 3e, we compare the movement energy loss for $\rho = 4$ and $\rho = 1$. As expected, $\rho = 1$ results in slightly lower GP-MD movement energy loss due to the tradeoff shifting towards the movement cost.

## 5 Conclusions

We have considered the problem of optimizing an unknown cost function subject to time-varying contextual information, as well as *movement costs* of changing the selected action from round to

[2]According to the power equation from [7], $E_M(x, x')$ would depend on $V_w(x, t)$, whereas, we assume our movement cost is based on a fixed metric and is independent of contexts. Hence, we simply approximate $V_w(x, t)$ by $V_r$ and consider time-independent movement costs.

round. Our problem formulation is motivated by Airborne Wind Energy systems, where one seeks to optimize the operating altitude of the wind turbine to maximize the amount of generated energy. We propose a novel algorithm, GP-MD, which makes use of GP confidence bounds and employs the mirror descent techniques from [19] for solving MTS problems. We analyze the theoretical performance of our algorithm by providing a rigorous regret bound. Moreover, we demonstrate its performance in synthetic experiments and on an AWE application by using real-world data. GP-MD carefully trades off service and movement costs while at the same time learning about the unknown objective function and yielding improved performance (i.e., generating more energy) compared to the considered baselines. Our setup and analysis open up multiple interesting directions for further exploration. For instance, an extension to continuous action spaces via discretization arguments is an immediate direction for future work. Another interesting direction is to analyze the single-episode setting and obtain general sublinear regret guarantees.

## 6 Acknowledgements

The authors would like to thank James R. Lee and Christian Coester for the various discussions regarding their paper [19] during the course of this work. This project has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme grant agreement No 815943.

## References

[1] Abbasi-Yadkori. Online learning for linearly parametrized control problems. 2013.

[2] Majid Abdolshah, Alistair Shilton, Santu Rana, Sunil Gupta, and Svetha Venkatesh. Cost-aware multi-objective Bayesian optimisation. *arXiv preprint arXiv:1909.03600*, 2019.

[3] Abdulelah Alkesaiberi, Fouzi Harrou, and Ying Sun. Efficient wind power prediction using machine learning methods: A comparative study. *Energies*, 2022.

[4] Leif Erik Andersson and Lars Imsland. Real-time optimization of wind farms using modifier adaptation and machine learning. *Wind Energy Science*, 2020.

[5] CJ Argue, Anupam Gupta, Ziye Tang, and Guru Guruganesh. Chasing convex bodies with linear competitive ratio. *Journal of the ACM (JACM)*, 2021.

[6] Ali Baheri and Chris Vermillion. Combined plant and controller design using batch bayesian optimization: a case study in airborne wind energy systems. *Journal of Dynamic Systems, Measurement, and Control*, 2019.

[7] Ali Baheri, Shamir Bin-Karim, Alireza Bafandeh, and Christopher Vermillion. Real-time control using Bayesian optimization: A case study in airborne wind energy systems. *Control Engineering Practice*, 2017.

[8] Yair Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. *Proceedings of 37th Conference on Foundations of Computer Science*, 1996.

[9] Philip Bechtle, Mark Schelbergen, Roland Schmehl, Udo Zillmann, and Simon Watson. Airborne wind energy resource analysis. *Renewable energy*, 2019.

[10] Ilija Bogunovic, Jonathan Scarlett, Andreas Krause, and Volkan Cevher. Truncated variance reduction: A unified approach to Bayesian optimization and level-set estimation. *Conference on Neural Information Processing Systems (NeurIPS)*, 2016.

[11] Allan Borodin, Nathan Linial, and Michael E Saks. An optimal on-line algorithm for metrical task system. *Journal of the ACM (JACM)*, 1992.

[12] Sébastien Bubeck, Yin Tat Lee, Yuanzhi Li, and Mark Sellke. Competitively chasing convex bodies. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, 2019.

[13] Sébastien Bubeck, Bo'az Klartag, Yin Tat Lee, Yuanzhi Li, and Mark Sellke. Chasing nested convex bodies nearly optimally. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2020.

[14] Sébastien Bubeck, Michael B Cohen, James R Lee, and Yin Tat Lee. Metrical task systems on trees via mirror descent and unfair gluing. *SIAM Journal on Computing*, 2021.

[15] Daniele Calandriello, Luigi Carratino, Alessandro Lazaric, Michal Valko, and Lorenzo Rosasco. Scaling Gaussian process optimization by evaluating a few unique candidates multiple times. *arXiv preprint arXiv:2201.12909*, 2022.

[16] Ian Char, Youngseog Chung, Willie Neiswanger, Kirthevasan Kandasamy, Andrew O Nelson, Mark Boyer, Egemen Kolemen, and Jeff Schneider. Offline contextual Bayesian optimization. *Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

[17] Sayak Ray Chowdhury and Aditya Gopalan. On kernelized multi-armed bandits. *International Conference on Machine Learning (ICML)*, 2017.

[18] Sayak Ray Chowdhury and Aditya Gopalan. Online learning in kernelized Markov decision processes. *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019.

[19] Christian Coester and James R Lee. Pure entropic regularization for metrical task systems. *Conference on Learning Theory (COLT)*, 2019.

[20] Christian Coester and James R Lee. Pure entropic regularization for metrical task systems. *arXiv preprint arXiv:1906.04270*, 2019.

[21] Bart M Doekemeijer, Daan C Van Der Hoek, and Jan-Willem van Wingerden. Model-based closed-loop wind farm control for power maximization using bayesian optimization: a large eddy simulation study. In *2019 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 2019.

[22] Dave Elliot. Flights of fancy: airborne wind turbines, 2014.

[23] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 2004.

[24] Jose Pablo Folch, Shiqiang Zhang, Robert M Lee, Behrang Shafei, David Walz, Calvin Tsay, Mark van der Wilk, and Ruth Misener. Snake: Bayesian optimization with pathwise exploration. *arXiv preprint arXiv:2202.00060*, 2022.

[25] Gautam Goel and Adam Wierman. An online algorithm for smoothed regression and lqr control. *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019.

[26] Gautam Goel, Yiheng Lin, Haoyuan Sun, and Adam Wierman. Beyond online balanced descent: An optimal algorithm for smoothed online optimization. *Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

[27] Javier González, Zhenwen Dai, Philipp Hennig, and Neil Lawrence. Batch Bayesian optimization via local penalization. *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016.

[28] Kirthevasan Kandasamy, Gautam Dasarathy, Jeff Schneider, and Barnabás Póczos. Multi-fidelity Bayesian optimisation with continuous approximations. *International Conference on Machine Learning (ICML)*, 2017.

[29] Tarun Kathuria, Amit Deshpande, and Pushmeet Kohli. Batched Gaussian process bandit optimization via determinantal point processes. *Conference on Neural Information Processing Systems (NeurIPS)*, 2016.

[30] Johannes Kirschner and Andreas Krause. Information directed sampling and bandits with heteroscedastic noise. *Conference On Learning Theory (COLT)*, 2018.

[31] Johannes Kirschner, Ilija Bogunovic, Stefanie Jegelka, and Andreas Krause. Distributionally robust Bayesian optimization. *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

[32] Andreas Krause and Cheng Soon Ong. Contextual Gaussian process bandit optimization. *Conference on Neural Information Processing Systems (NeurIPS)*, 2011.

[33] Remi Lam and Karen Willcox. Lookahead Bayesian optimization with inequality constraints. *Conference on Neural Information Processing Systems (NeurIPS)*, 2017.

[34] Remi Lam, Karen Willcox, and David H Wolpert. Bayesian optimization with a finite budget: An approximate dynamic programming approach. *Conference on Neural Information Processing Systems (NeurIPS)*, 2016.

[35] Chi-Heng Lin, Joseph D Miano, and Eva L Dyer. Bayesian optimization for modular black-box systems with switching costs. *Uncertainty in Artificial Intelligence (UAI)*, 2021.

[36] Minghong Lin, Adam Wierman, Lachlan LH Andrew, and Eno Thereska. Dynamic right-sizing for power-proportional data centers. *IEEE/ACM Transactions on Networking*, 2012.

[37] Yiheng Lin, Gautam Goel, and Adam Wierman. Online optimization with predictions and non-convex losses. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2020.

[38] Jonas Močkus. On Bayesian methods for seeking the extremum. In *Optimization Techniques*, 1975.

[39] Nikolaos Moustakis, Sebastiaan Paul Mulders, Jens Kober, and Jan-Willem van Wingerden. A practical bayesian optimization approach for the optimal estimation of the rotor effective wind speed. In *2019 American Control Conference (ACC)*. IEEE, 2019.

[40] Jinkyoo Park. Contextual Bayesian optimization with trust region (cbotr) and its application to cooperative wind farm control in region 2. *Sustainable Energy Technologies and Assessments*, 2020.

[41] Jinkyoo Park and Kincho H Law. A bayesian optimization approach for wind farm power maximization. In *Smart Sensor Phenomena, Technology, Networks, and Systems Integration 2015*. International Society for Optics and Photonics, 2015.

[42] C. E. Rasmussen and C. K. Williams. Gaussian processes for machine learning. *volume 1. MIT press Cambridge*, 2006.

[43] Mark Sellke. Chasing convex bodies optimally. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2020.

[44] Pier Giuseppe Sessa, Ilija Bogunovic, Maryam Kamgarpour, and Andreas Krause. No-regret learning in unknown games with correlated payoffs. *Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

[45] Pier Giuseppe Sessa, Ilija Bogunovic, Maryam Kamgarpour, and Andreas Krause. Mixed strategies for robust optimization of unknown objectives. *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

[46] Pier Giuseppe Sessa, Ilija Bogunovic, Maryam Kamgarpour, and Andreas Krause. Learning to play sequential games versus unknown opponents. *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[47] Pier Giuseppe Sessa, Ilija Bogunovic, Andreas Krause, and Maryam Kamgarpour. Contextual games: Multi-agent learning with side information. *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[48] Guanya Shi, Yiheng Lin, Soon-Jo Chung, Yisong Yue, and Adam Wierman. Online optimization with memory and competitive control. *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[49] Jialin Song, Yuxin Chen, and Yisong Yue. A general framework for multi-fidelity Bayesian optimization with Gaussian processes. *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019.

[50] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *International Conference on Machine Learning (ICML)*, 2010.

[51] Kübra Yazici and Semra Boran. Short-term wind power prediction approach based on bayesian optimization and ensemble learning. *Journal of Intelligent Systems: Theory and Applications*.

## Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 5
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Added in supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Internal Cluster was used
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

# Supplementary Material

## Movement Penalized Bayesian Optimization with Application to Wind Energy Systems

## A Metrical Task Systems (MTS)

Let $(\mathcal{X}, d)$ be a finite metric space with $|\mathcal{X}| = n > 1$ as defined in Section 2. Henceforth, we denote the points in $\mathcal{X}$ as $\{x^1, \ldots, x^n\}$. The MTS problem runs over a single episode of horizon $T$. At every time instant $1 \leq t \leq T$, the learner receives a non-negative cost function over $\mathcal{X}$, $c_t : \mathcal{X} \to \mathbb{R}_+$ corresponding to each point in $\mathcal{X}$. The goal of any online optimization algorithm in this setting is to choose $x_t \in \mathcal{X}$ such that both the cost incurred over the rounds and sum over the distances between its subsequent decisions as measured by $d(x_t, x_{t-1})$ is minimized. We do not include episode $m$ in the variables' subscripts as it runs over a single episode and instead include the time horizon $T$. Hence, $D_T = \{x_1, x_2, \ldots, x_T\}$ denotes the action sequence of length $T$ outputted by an algorithm and $S_T(D_T)$ and $M_T(D_T)$ (Eq. (2)) denote the corresponding service and movement costs respectively. Then the total cost incurred by such an algorithm for initial state $x_0$ up to a time horizon $T$ is

$$\text{cost}_T(D_T) = \sum_{t=1}^{T} c_t(x_t) + d(x_t, x_{t-1}).$$

Next, we recall some notions about competitive ratio for MTS from [19] which are useful to prove our regret guarantees in Appendix E. Here, $D_T^*$ denotes the offline optimal sequence which minimizes $\text{cost}_T(D_T)$. Here we note that this offline optimal sequence $D_T^*$ depends on the initial state $x_0$.
**Competitive Ratio**: If there exist constants $\alpha, \beta$ such that for every cost sequence $(c_t)_{t=1}^T$, arbitrary initial state $x_0 \in \mathcal{X}$ and distance metric $d(\cdot, \cdot)$,

$$\text{cost}_T(D_T) \quad \leq \quad \alpha \text{cost}_T(D_T^*) + \beta,$$

then the algorithm is $\alpha$-competitive.
**Refined Competitive Ratio Guarantees**: If there exist constants $\alpha, \alpha', \beta, \beta'$ such that for every cost sequence $(c_t)_{t=1}^T$, arbitrary initial state $x_0 \in \mathcal{X}$ and distance metric $d(\cdot, \cdot)$,

$$S_T(D_T) \leq \alpha \text{cost}_T(D_T^*) + \beta, \tag{18}$$

$$M_T(D_T) \leq \alpha' \text{cost}_T(D_T^*) + \beta', \tag{19}$$

then the algorithm is $\alpha$-competitive for service costs and $\alpha'$-competitive for movement costs.

## B Hierarchically Separated Tree (HST) Metric

We define the tree $\mathcal{T} = (V, E)$ with the root vertex being $r$ and weight corresponding to each vertex $v \in V$ being $w_v$. Let $\mathcal{L}$ denote the set of leaves in this tree $\mathcal{T}$. Consider the case when the weights are non-increasing while moving from root to any leaf. We assign the edge from any vertex $u$ to its parent $par(u)$ the weight $w_u$.
**Distance Function**: We define the distance metric $d_{\mathcal{T}}(l, l')$ between any two leaves $l, l'$ in the tree as the weighted length of the path from $l$ to $l'$. For example, as shown in Fig. 4, Then the space with states $\mathcal{L}$ and distance metric $d_{\mathcal{T}}$ is defined as the HST metric space denoted by $(\mathcal{L}, d_{\mathcal{T}})$ ([19]).

### B.1 $\tau$-HST Metric

In an HST Metric space if the weights are exponentially decreasing, i.e., $w_u \leq w_{par(u)}/\tau$ then we call it a $\tau$-HST Metric. Such metric spaces are of particular interest due to a result from [8] which states that any online algorithm which is $\mathcal{O}(g(n))$-competitive for $\tau$-HST metric space is $\mathcal{O}(g(n) \log n)$-competitive for arbitrary $n$-point metric spaces where $g(n)$ is some function on $n$.

### B.2 FRT Algorithm

The FRT algorithm from [23] is a randomized algorithm and outputs a tree $\mathcal{T}$ whose leaves correspond to points in $\mathcal{X}$ but the tree distance between any two points (leaves) $x^i, x^j \in \mathcal{X}$ (or $\mathcal{L}$) is $d_{\mathcal{T}}(x^i, x^j)$

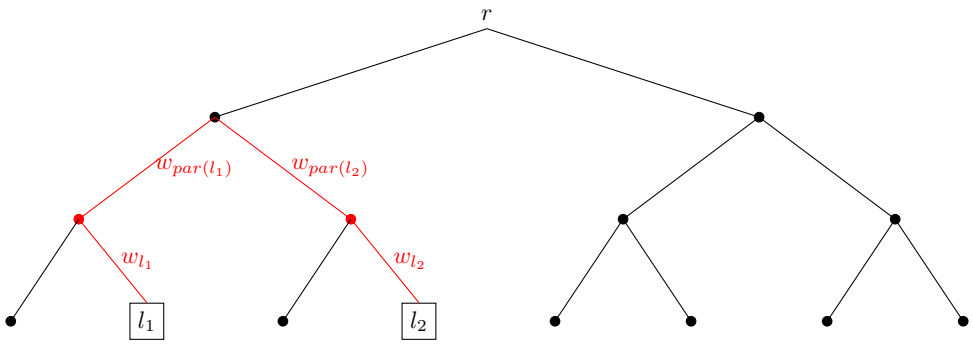



Figure 4: $d_{\mathcal{T}}(l_1, l_2) = w_{l_1} + w_{par(l_1)} + w_{l_2} + w_{par(l_2)}$.

and satisfies for any $x^i, x^j \in \mathcal{X}$

$$\mathbb{P}[d_{\mathcal{T}}(x^i, x^j) \geq d(x^i, x^j)] = 1, \tag{20}$$

$$\mathbb{E}_{\mathcal{T}}\left[d_{\mathcal{T}}(x^i, x^j)\right] \leq \mathcal{O}(\log n) d(x^i, x^j), \tag{21}$$

where $d(\cdot, \cdot)$, is the original metric and expectation is w.r.t. the random tree $\mathcal{T}$ generated by the FRT algorithm [23].

# C Action Representation

This section is intended to explain in detail about the randomized algorithm setup for MTS as elucidated in [19] and [14]. But instead of using the cost sequence $(c_t)_{t=1}^T$, we explain in terms of $f(\cdot, e_t)$ which is more relevant to our setting. In our proof, we only consider episodic expectation (see Eq. (46)) and want to understand how the randomized algorithm evolves within a single episode. Hence similar to MTS setup from Coester and Lee [19] we detail this randomization section for a single episode.

## C.1 Action Randomization

Let $\mathcal{P}(\mathcal{X})$ be the set of probability measures supported on $\mathcal{X}$. For $\mu, \nu \in \mathcal{P}(\mathcal{X})$ denote $\mathbb{W}^1(\mu, \nu)$ as the Wasserstein-1-distance between $\mu$ and $\nu$ defined based on $d(\cdot, \cdot)$. A randomized online algorithm at time $t$ outputs a random action $x_t \sim p_t$ where probability distribution $p_t \in \mathcal{P}(\mathcal{X})$. As defined earlier, we denote this random output sequence as $D_T = \{x_1, x_2, \ldots, x_T\}$. In this randomized setting it is more intuitive to consider the *expected cost*.

The *expected movement cost* for distance metric $d(\cdot, \cdot)$ is defined as follows:

$$\mathbb{E}\left[M_T(D_T)\right] = \mathbb{E}\left[\sum_{t=1}^{T} d(x_{t-1}, x_t)\right] = \sum_{t=1}^{T} \mathbb{E}[d(x_{t-1}, x_t)].$$

Here note that each term in the sum $\mathbb{E}[d(x_{t-1}, x_t)]$ is expectation w.r.t. a joint distribution of $(x_{t-1}, x_t)$. Under certain assumptions on the sampling process (described in Appendix C.2), the joint distribution becomes

$$\mathbb{E}\left[M_T(D_T)\right] = \sum_{t=1}^{T} \mathbb{W}^1(p_{t-1}, p_t), \tag{22}$$

And the *expected service cost* is then,

$$\mathbb{E}\left[S_T(D_T)\right] = \mathbb{E}\left[\sum_{t=1}^{T} f(x_t, e_t)\right] = \sum_{t=1}^{T} \langle f(\cdot, e_t), p_t \rangle, \tag{23}$$

$$\mathbb{E}\left[\text{cost}_T(D_T)\right] = \sum_{t=1}^{T} \mathbb{W}^1(p_{t-1}, p_t) + \sum_{t=1}^{T} \langle f(\cdot, e_t), p_t \rangle. \tag{24}$$

An randomized online algorithm is said to be $\alpha$-*competitive* if for all $x_0 \in \mathcal{X}$ and any $f$, it outputs a sequence $D_T$ whose expected cost for some $\beta > 0$ satisfies the following condition w.r.t. the offline optimal sequence $D_T^*$ for some metric $d(\cdot, \cdot)$:

$$\mathbb{E}\left[\text{cost}_T(D_T)\right] \leq \alpha \text{cost}_T(D_T^*) + \beta.$$

### C.2 Sampling from Joint Distribution

Since our goal is to minimize the total cost we choose a joint distribution $\zeta_t \in \Pi(p_{t-1}, p_t)$ which minimizes the expected cost as follows:

$$\mathbb{E}_{\zeta_t}[d(x_{t-1}, x_t)] = \inf_{\zeta \in \Pi(p_{t-1}, p_t)} \mathbb{E}_{\zeta}[d(U_{t-1}, U_t)],$$

where $\Pi(p_{t-1}, p_t)$ denotes the set of all random variables $(U_{t-1}, U_t)$ whose marginals are $p_{t-1}$ and $p_t$, respectively. By the definition of Wasserstein-1 distance where the Wasserstein cost function is assumed to be $d(\cdot, \cdot)$ we have,

$$\inf_{\zeta \in \Pi(p_{t-1}, p_t)} \mathbb{E}_{\zeta}[d(U_{t-1}, U_t)] = \mathbb{W}^1(p_{t-1}, p_t). \tag{25}$$

Hence, we want to ensure that the subsequent actions in algorithm 1 $(x_{t-1}, x_t)$ follows the distribution $\zeta_t$ making our total movement cost

$$\mathbb{E}\left[M_T(D_T)\right] = \sum_{t=1}^{T} \mathbb{W}^1(p_{t-1}, p_t). \tag{26}$$

In order to achieve this, we take the following steps. We first note that the initial action $x_0$ is given. Hence, after obtaining $p_1$ using Mirror Descent procedure as done in Line-11 of Algorithm 1, we estimate $\zeta_1$, then calculate the conditional distribution $\zeta_1(U_1|U_0 = x_0)$, and sample $x_1$ from this conditional distribution. At any future time instant $t$, after obtaining $p_t$, we repeat this process and calculate the conditional distribution $\zeta_t(U_t|U_{t-1} = x_{t-1})$ and sample $x_t$ from it (Line 12 of Algorithm 1). In this way we ensure that at each time instant $t$, $(x_{t-1}, x_t)$ is sampled from $\zeta_t$ and hence attaining the movement cost as sum of Wasserstein-1 distances.

### C.3 Randomized Action Representation in $\tau$-HST

In this section, we explain the randomized action representation in $\tau$-HST space introduced in Section 3 in further detail. Also, we show the Wasserstein-1 distance (Eq. (25)) can be simplified in the $d_{\mathcal{T}}$-metric when the actions are leaves of $\mathcal{T}$ as done in [14] and [19].

From the analysis in the previous section, in order to calculate the movement cost, we need to compute the Wasserstein-1 distances between 2 probability distributions over the leaves of the constructed tree. As we now consider $\tau$-HST metric space, the distance metric is $d_{\mathcal{T}}$ and Wasserstein-1 distance is denoted as $\mathbb{W}^1_{\mathcal{T}}(\cdot, \cdot)$. We first recall the following representation of the randomized action described in Section 3 from [14]. This will be useful in both the Wasserstein-1 distance calculation and in Algorithm 1.

Let $\mathcal{T} = (V, E)$ be a tree with vertices $V$, edges $E$, root $r$ and leaves $\mathcal{L}$. We define a convex polytope on the space $\mathbb{R}_+^V$

$$K_{\mathcal{T}} := \Big\{ z \in \mathbb{R}_+^{|V|} : z_r = 1, z_u = \sum_{\nu \in \mathcal{C}(u)} z_\nu \quad \forall u \in V \backslash \mathcal{L} \Big\},$$

where $\mathcal{C}(u)$ denotes the children of $u$. Note that by the above definition for any $z \in K_{\mathcal{T}}$, it holds that

$$\sum_{l \in \mathcal{L}} z_l = 1.$$

And the Wasserstein-1 distance between 2 random actions in $(\mathcal{L}, d_{\mathcal{T}})$ specified by the probability distributions $l(z)$ and $l(z')$ is as follows:

$$\mathbb{W}^1_{\mathcal{T}}(l(z), l(z')) := \sum_{u \in V} w_u |z_u - z'_u| = ||z - z'||_{l_1(w)}. \tag{27}$$

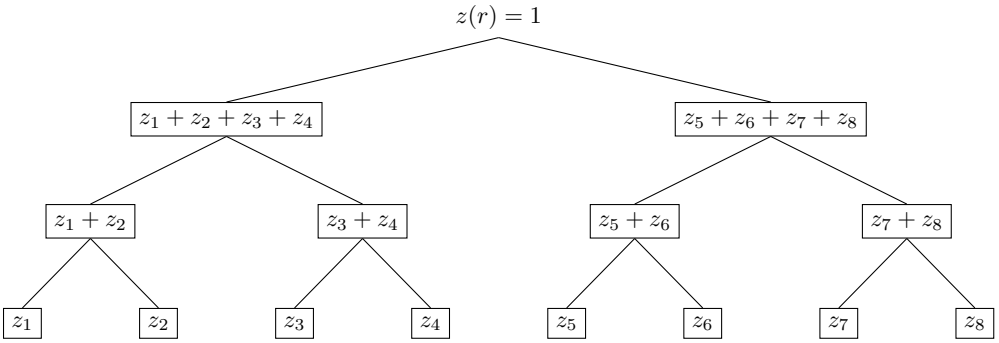
where $w_u$ are the weights of the edges from $u$ to $par(u)$ in $\mathcal{T}$. Hence the total cost w.r.t. $d_{\mathcal{T}}(\cdot,\cdot)$-metric denoted by $\text{cost}_T^{\mathcal{T}}(D_T)$ (Eq. (24)) now becomes

$$\mathbb{E}\left[\text{cost}_T^{\mathcal{T}}(D_T)\right] = \sum_{t=1}^{T} ||z_{t-1} - z_t||_{l_1(w)} + \sum_{t=1}^{T} \langle f(\cdot, e_t), l(z_t)\rangle. \tag{28}$$

Hence, for the $\tau$-HST metric space $(\mathcal{L}, d_{\mathcal{T}})$, where leaves correspond to actions, $z$ defines a probability distribution over all the states. Each entry $z_u$ represents the probability that the selected action $x$ belongs to the leaves of the subtree rooted at $u$, i.e., $z_u = \mathbb{P}(x \in \mathcal{L}(u))$. Also, We note that $z$ is completely defined when all $z_l \in \mathcal{L}$ is provided. And for a deterministic state $x \in \mathcal{X}$, the corresponding state in $K_{\mathcal{T}}$ is

$$z_l = \left\{ \begin{array}{ll} 1, & \text{for } l = x \\ 0, & \text{for } l \neq x \end{array} \right\} \quad \forall l \in \mathcal{L}, \qquad z_u = \left\{ \begin{array}{ll} 1, & \text{for } x \in \mathcal{L}(u) \\ 0, & \text{for } x \notin \mathcal{L}(u) \end{array} \right\} \quad \forall u \in V \backslash \mathcal{L} \tag{29}$$

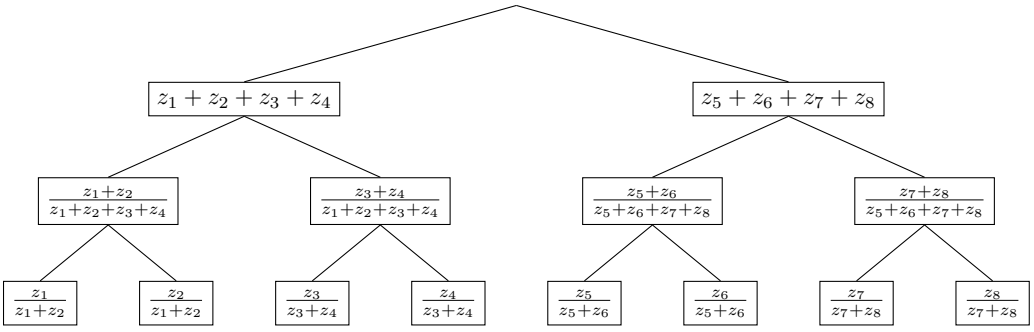
We can visualize this polytope with the example figure:




### C.4 Action Representation using Conditional Probabilities

In Section 3, we provided an intutive understanding of how the algorithm works based on the calculation of $q^{(u)}$ corresponding to each internal vertex $u$ (Eq. (13)). We also discussed how this can be viewed as a conditional probability $\mathbb{P}(x \in \mathcal{L}(\nu) | x \in \mathcal{L}(u))$ for $\nu \in \mathcal{C}(u)$ (children of $u$) and how it can be used to calculate the actual probabilities over the actions-$l(z)$ (Eq. (15)).

We first begin with a visualization of this $q$ based on the visualization in Appendix C.3 and Eq. (15):




We explain why working in terms of $q$ rather than $z$ is beneficial. The Bregman divergence $D^{(u)}$ in the mirror descent update (Eq. (13)) uses a potential function and the authors of [19] observed that the conditional probability based potential function imitates the weighted entropy of the probability distribution over the leaves of a $\tau$-HST tree. The intuition for the above statement from [19] is described next.

We first recall $Q_{\mathcal{T}}$ and $Q_{\mathcal{T}}^{(u)}$ from Section 3. $Q_{\mathcal{T}}$ is the set of valid conditional probabilities

$$Q_{\mathcal{T}} := \Big\{ q \in \mathcal{R}_+^{|V \backslash r|} : \sum_{\nu \in \mathcal{C}(u)} q_\nu = 1 \quad \forall u \in V \backslash \mathcal{L} \Big\}. \tag{30}$$

Moreover, given $q \in Q_{\mathcal{T}}$ we define the vector $q^{(u)} := [q_\nu,\ \nu \in \mathcal{C}(u)]$ as the conditional distribution over children of $u$, and let $Q_{\mathcal{T}}^{(u)}$ be the set of all valid distributions $q^{(u)}$.

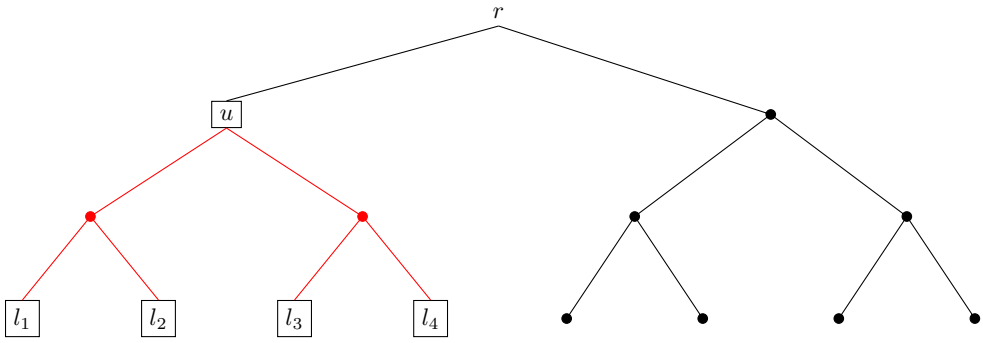
We define the potential function $\Phi^{(u)}$ used for Bregman Divergence $D^{(u)}$ in the mirror descent update (Eq. (13)) for $q \in Q_{\mathcal{T}}$ and the corresponding $q^{(u)} \in Q_{\mathcal{T}}^{(u)}$ for a particular vertex $u \in V$ as follows:

$$\Phi^{(u)}(q^{(u)}) := \frac{1}{\kappa} \sum_{\nu \in \mathcal{C}(u)} \frac{w_\nu}{\eta_\nu} (q_\nu^{(u)} + \delta_\nu) \log(q_\nu^{(u)} + \delta_\nu), \tag{31}$$

where

$$\begin{aligned} \theta_u &:= \frac{|\mathcal{L}(u)|}{|\mathcal{L}(par(u))|}, \\ \eta_u &:= 1 + \log(1/\theta_u), \\ \delta_u &:= \frac{\theta_u}{\eta_u}, \\ \kappa &\geq 1 \quad \text{(fixed constant for all } u\text{)}. \end{aligned}$$

**Potential Function Intuition.** In this section we elucidate how the potential function defined above Eq. (31) can be viewed as an approximate weighted entropy of a probability distribution over the leaves of a $\tau-$HST tree. Recalling our definitions, we have $\mathcal{T} = (V, E)$, a tree with vertices $V$, edges $E$ and leaves $\mathcal{L}$. $Y$ is a random variable with support $\mathcal{L}$. Let $\varepsilon_u$ denote the event that $\{Y \in \mathcal{L}(u)\}$ where $\mathcal{L}(u)$ is the set of leaves under the tree rooted at $u$. For example, from the picture below, $\mathcal{L}(u) = \{l_1, l_2, l_3, l_4\}$




We calculate the entropy of such a random variable $Y$ and connect it with the potential function Eq. (31). Using the definition of entropy,

$$H(Y) = \sum_{l \in \mathcal{L}} P[\varepsilon_l] \log \frac{1}{P[\varepsilon_l]} \tag{32}$$

$$= \sum_{l \in \mathcal{L}} P[\varepsilon_l] \log \frac{P[\varepsilon_{par(l)}]}{P[\varepsilon_l] P[\varepsilon_{par(l)}]} \tag{33}$$

$$= \sum_{l \in \mathcal{L}} P[\varepsilon_l] \log \frac{P[\varepsilon_{par(l)}]}{P[\varepsilon_l]} + \sum_{l \in \mathcal{L}} P[\varepsilon_l] \log \frac{1}{P[\varepsilon_{par(l)}]} \tag{34}$$

$$= \sum_{l \in \mathcal{L}} P[\varepsilon_l] \log \frac{P[\varepsilon_{par(l)}]}{P[\varepsilon_l]} + \sum_{u \in par(\mathcal{L})} \sum_{l \in \mathcal{C}(u)} P[\varepsilon_l] \log \frac{1}{P[\varepsilon_u]} \tag{35}$$

$$= \sum_{l \in \mathcal{L}} P[\varepsilon_l] \log \frac{P[\varepsilon_{par(l)}]}{P[\varepsilon_l]} + \sum_{u \in par(\mathcal{L})} P[\varepsilon_u] \log \frac{1}{P[\varepsilon_u]} \tag{36}$$

$$= \sum_{l \in \mathcal{L}} P[\varepsilon_l] \log \frac{P[\varepsilon_{par(l)}]}{P[\varepsilon_l]} + \sum_{u \in par(\mathcal{L})} P[\varepsilon_u] \log \frac{P[\varepsilon_{par(u)}]}{P[\varepsilon_u]} + \sum_{u \in par(par(\mathcal{L}))} P[\varepsilon_u] \log \frac{1}{P[\varepsilon_u]} \tag{37}$$

$$= \sum_{u \in V \setminus \{r\}} P[\varepsilon_u] \log \frac{P[\varepsilon_{par(u)}]}{P[\varepsilon_u]}. \tag{38}$$

Here the transition from Eq. (36) to Eq. (37) is obtained from simplifying $\sum_{u \in par(\mathcal{L})} P[\varepsilon_u] \log \frac{1}{P[\varepsilon_u]}$ as we simplified $\sum_{l \in \mathcal{L}} P[\varepsilon_l] \log \frac{1}{P[\varepsilon_l]}$. Applying the previous argument recursively we obtain the final equation (Eq. (38)). From Eq. (38) the weighted version of the entropy can be defined as

$$H(Y, w) = \sum_{u \in V \setminus \{r\}} w_u P[\varepsilon_u] \log \frac{P[\varepsilon_{par(u)}]}{P[\varepsilon_u]}. \tag{39}$$

Note that from definition of $z \in K_{\mathcal{T}}$ in Appendix C.3, $z_u$ and $z_{par(u)}$ are analogous to $P[\varepsilon_u]$ and $P[\varepsilon_{par(u)}]$ respectively. Using this and a negative weighted version of Eq. (39) (inversion inside $\log$) the authors of [19] define the following regularizer for $z \in K_{\mathcal{T}}$.

$$\Phi(z) := \sum_{u \in V \setminus \{r\}} \frac{w_u}{\eta_u} (z_u + \delta_u z_{par(u)}) \log(\frac{z_u}{z_{par(u)}} + \delta_u). \tag{40}$$

Here $\delta_u$ signifies noise added to entropy calculation and ensures fast updates of $\frac{z_u}{z_{par(u)}}$ when it is close to zero. $\eta_u$ signifies the relative importance given to present costs at time $t$ vs past costs observed upto time $t-1$ (For further details refer to Coester and Lee [19, Section 1.3]).

From Eq. (15), we know that $q_u$ is analogous to $\frac{z_u}{z_{par(u)}}$. Based on Eq. (40) and replacing $q_u$, the authors of [19] define potential function for $q^{(u)} \in Q_{\mathcal{T}}^{(u)}$ corresponding to $q \in Q_{\mathcal{T}}$ for a particular $u$ as follows:

$$\Phi^{(u)}(q^{(u)}) := \frac{1}{\kappa} \sum_{\nu \in \mathcal{C}(u)} \frac{w_\nu}{\eta_\nu} (q_\nu + \delta_\nu) \log(q_\nu + \delta_\nu).$$

# D Algorithm

In this section we first provide some insights from [20, Sectiom 2.1] to solve the optimization problem in Eq. (13). Then we proceed to practically illustrate the flow of the algorithm from leaves to root w.r.t. $q \in Q_{\mathcal{T}}$ calculations using Eq. (13).

## D.1 Divergence and Optimization Calculations

In order to solve the optimization problem Eq. (13), we first need to calculate the Bregman Divergence $D^{(u)}$. The Bregman Divergence for some potential function $\Phi(\cdot)$ is defined as follows:

$$D_\Phi(y||x) := \Phi(y) - \Phi(x) - \langle \nabla \Phi(x), y - x \rangle. \tag{41}$$

In our case, Bregman Divergence $D^{(u)}(\cdot||\cdot)$ calculates the divergence between two conditional probability vectors $q^{(u)}, q'^{(u)} \in Q_{\mathcal{T}}^{(u)}$ defined over the children of vertex $u$ w.r.t. the potential function $\Phi^{(u)}$. Here $q, q' \in Q_{\mathcal{T}}$ (Eq. (30)) and potential function $\Phi^{(u)}$ is as defined in Eq. (31). Hence we have,

$$D^{(u)}(q^{(u)}||q'^{(u)}) := \frac{1}{\kappa} \sum_{\nu \in \mathcal{C}(u)} \frac{w_\nu}{\eta_\nu} \left[ (q_\nu + \delta_\nu) \log(\frac{q_\nu + \delta_\nu}{q'_\nu + \delta_\nu}) + q'_\nu - q_\nu \right].$$

Now the authors of [19] use KKT conditions and Lagrange multipliers to solve Eq. (13) by substituting the definition of Bregman Divergence in Eq. (41). It yields that the solution to Eq. (13) for $q'^{(u)} = q_h^{(u)}$, $q^{(u)} = q_{h-1}^{(u)}$ and $e = e_{h,m}$ satisfies

$$\nabla \Phi^{(u)}(q'^{(u)}) = \nabla \Phi^{(u)}(q^{(u)}) - \text{lcb}_m^{(u)}(\cdot, e) - \beta^{(u)} - \alpha^{(u)}. \tag{42}$$

Here $\beta^{(u)}$ and $\alpha^{(u)}$ are the Lagrange multipliers for the constraints in $Q_{\mathcal{T}}^{(u)}$ to ensure that for any $q^{(u)} \in Q_{\mathcal{T}}^{(u)}$, $q^{(u)}$ is actually a probability vector and comprises of the following 2 constraints,

$$\sum_{\nu \in \mathcal{C}(u)} q_\nu^{(u)} = 1 \qquad \text{and} \quad q_\nu^{(u)} \geq 0 \quad \text{for} \quad \nu \in \mathcal{C}(u).$$

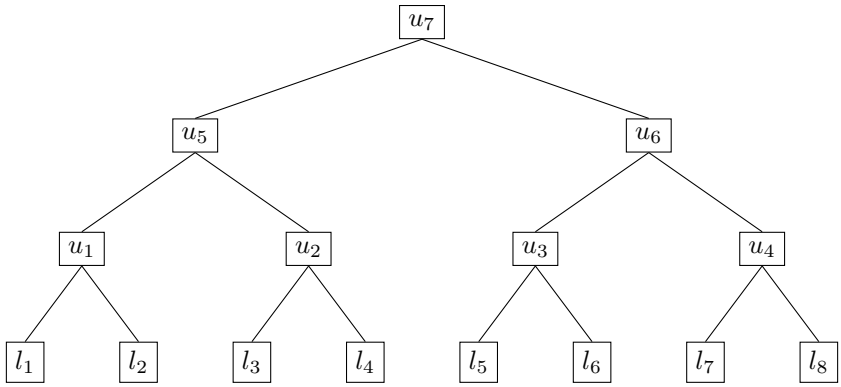



Figure 5: Example Tree $\mathcal{T}$

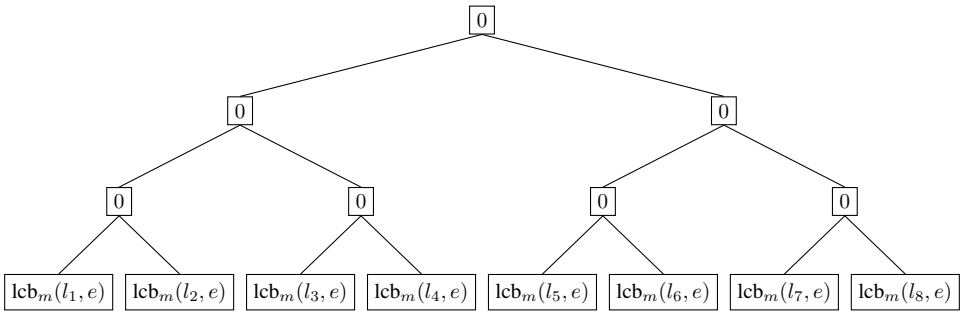



Figure 6: Cost of each vertex $u \in V$ after initialization in Line-6 of Algorithm 1 ($e = e_{h,m}$).

Now calculating the gradient of the potential function in Eq. (31) we have,

$$\left(\nabla \Phi^{(u)}(q^{(u)})\right)_\nu = \frac{1}{\kappa}\frac{w_\nu}{\eta_\nu}(1 + \log(q_\nu + \delta_\nu)).$$

Substituting this in Eq. (42), the solution to Eq. (13) for $q'^{(u)} = q_h^{(u)}$, $q^{(u)} = q_{h-1}^{(u)}$ and $e = e_{h,m}$ will be

$$q_\nu'^{(u)} = (q_\nu^{(u)} + \delta_\nu)\exp\{\kappa\frac{\eta_\nu}{w_\nu}(\beta^{(u)} - (\text{lcb}_m^{(u)}(\nu, e) - \alpha_\nu))\} - \delta_\nu. \tag{43}$$

Eq. (43) can be solved in polynomial time using interior point methods. But for practical purposes, we use projected gradient descent w.r.t. $\alpha$ to solve this problem.

### D.2 Illustration

In this section we illustrate the subroutine in lines $6-9$ of Algorithm 1. For the tree $\mathcal{T}$ with vertices $V$ and leaves $\mathcal{L}$ shown in Fig. 5, let the topological ordering $\mathcal{OD}(V\backslash\mathcal{L})$ (Line 7 in Algorithm 1) of internal vertices $V\backslash\mathcal{L}$ be $\{u_1, u_2, u_3, u_4, u_5, u_6, u_7\}$. Note that the ordering ensures that every child in $\mathcal{T}$ occurs before its parent. Algorithm 1 runs the mirror descent update Eq. (13) according to the order mentioned in $\mathcal{OD}(V\backslash\mathcal{L})$.

We first illustrate the state of the costs corresponding to each vertex $v \in V$ after the initialization in Line-6 of Algorithm 1 through Fig. 6 for $e = e_{h,m}$.

Once the sub-routine in Lines $8-9$ is run for $u_{j=1,2,3,4}$ in $\mathcal{OD}(V\backslash\mathcal{L})$ one obtains the conditional probabilities $q_l^{(par(l))}$ for all leaves $l \in \mathcal{L}$ using Eq. (13) and costs for internal vertices $\{u_1, u_2, u_3, u_4\}$ denoted by $\text{lcb}_m(u_j, e_{h,m})$ using Eq. (14). In Fig. 7 we depict this calculated costs and probabilities.

Then the sub-routine in Lines $8-9$ is run for $u_{j=5,6}$ in $\mathcal{OD}(V\backslash\mathcal{L})$ and one obtains the conditional probabilities $q_{u_j}^{(par(u_j))}$ for $j = \{1, 2, 3, 4\}$ using Eq. (13) and costs for internal vertices $\{u_5, u_6\}$ denoted by $\text{lcb}_m(u_j, e_{h,m})$ using Eq. (14). In Fig. 8 we depict this calculated costs and probabilities.

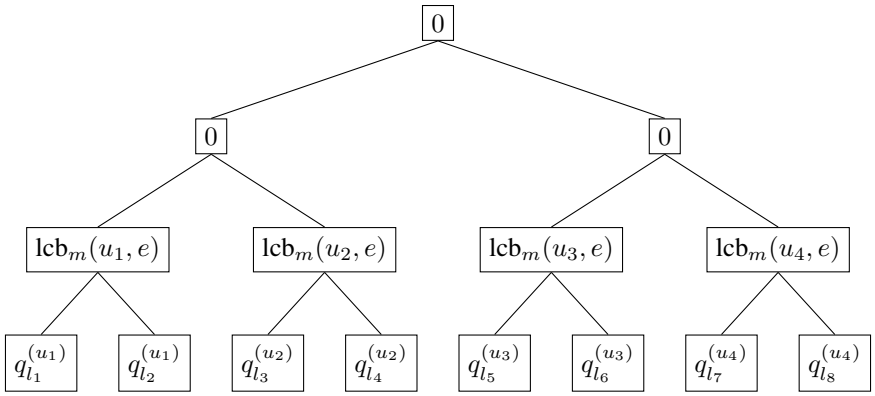



Figure 7: Conditional Probabilities for $\{l_1, \ldots, l_8\}$ and costs for $\{u_1, \ldots, u_4\}$ ($e = e_{h,m}$)

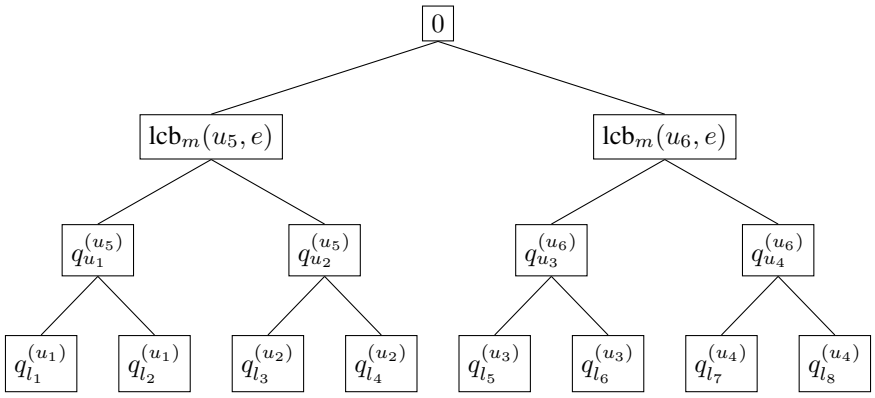



Figure 8: Conditional Probabilities for $\{u_1, \ldots, u_4\}$ and costs for $\{u_5, u_6\}$ ($e = e_{h,m}$)

Finally the sub-routine in Lines $8 - 9$ is run for $u_{j=7}$ in $\mathcal{OD}(V \backslash \mathcal{L})$ and one obtains the conditional probabilities $q_{u_j}^{(par(u_j))}$ for $j = \{5, 6\}$ using Eq. (13) and costs for internal vertex $\{u_7\}$ denoted by $\text{lcb}_m(u_7, e_{h,m})$ using Eq. (14) (also signifies the expected cost if the action is sampled from the calculated probabilities after conversion using Eq. (15)). In Fig. 9 we depict this calculated costs and probabilities.

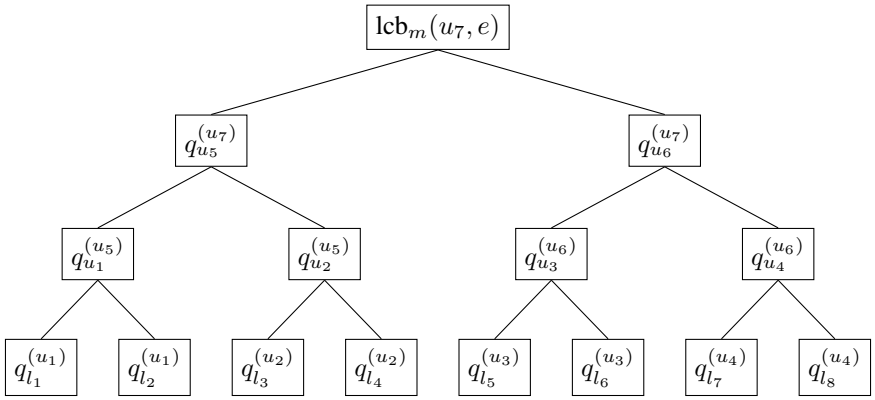



Figure 9: Conditional Probabilities for $\{u_5, u_6\}$ and cost for $\{u_7\}$ ($e = e_{h,m}$)

## E Proof of Thm. 1

The first step in the proof of Thm. 1 is to rewrite the cumulative episodic regret (Eq. (4)) as a sum of expected episodic regret conditioned w.r.t. data observed till the previous episode. In particular, the main idea is to upper bound $R^{\alpha,\beta}_{N_{ep}} = \sum_{m=1}^{N_{ep}} r^{\alpha,\beta}_m$ by $\sum_{m=1}^{N_{ep}} \mathbb{E}_m[r^{\alpha,\beta}_m|\mathcal{F}_{m-1}]$. To do this, we make use of [30, Lemma 13] as we explain below. Here, the expectation $\mathbb{E}_m$ is w.r.t. the actions in episode $m$, $D_m = (x_{h,m})_{h=1}^H$ outputted by Algorithm 1, and $\mathcal{F}_m$ denotes the data collected by Algorithm 1 during the first $m$ episodes, i.e.,

$$\mathcal{F}_m = \{(x_{h,i}, e_{h,i}, y_{h,i})_{h=1}^H\}_{i=1}^m. \tag{44}$$

We note that in the episodic regret

$$r^{\alpha,\beta}_m = \sum_{h=1}^H f(x_{h,m}, e_{h,m}) + \sum_{h=1}^H d(x_{h,m}, x_{h-1,m}) - \alpha \cdot \mathrm{cost}_m(D^*_m) - \beta,$$

the term $\alpha \cdot \mathrm{cost}_m(D^*_m) - \beta$ is constant w.r.t. the expectation $\mathbb{E}_m$ as it is independent of the actions $(x_{h,m})_{h=1}^H$. Also, the episodic cost $\mathrm{cost}_m(D_m) = \sum_{h=1}^H f(x_{h,m}, e_{h,m}) + \sum_{h=1}^H d(x_{h,m}, x_{h-1,m})$ is trivially upper bounded by $H(B+\psi)$. This is because $f(x,e) \le B$ (follows from our assumptions $\|f\|_k \le B$ and $k(\cdot,\cdot) \le 1$) and $d(x,x') \le \psi$ as detailed in Section 2.

Then, according to [30, Lemma 13], with probability at least $1-\delta$ the cumulative regret from Eq. (4) is bounded as:

$$R^{\alpha,\beta}_{N_{ep}} = \sum_{m=1}^{N_{ep}} r^{\alpha,\beta}_m \tag{45}$$

$$\le \sum_{m=1}^{N_{ep}} \mathbb{E}_m[r^{\alpha,\beta}_m|\mathcal{F}_{m-1}] + 4H(B+\psi)\log\Big(\frac{4\pi^2 N_{ep}^2}{3\delta}\big(\log(N_{ep})+1\big)\Big). \tag{46}$$

Above, we have applied [30, Lemma 13] to the cumulative sum of the stochastic process $r^{\alpha,\beta}_m$ and used the fact that each episodic cost $\mathrm{cost}_m(D_m)$ is bounded by $0 \le \mathrm{cost}_m(D_m) \le H(B+\psi)$.

In what follows, we focus on bounding

$$\sum_{m=1}^{N_{ep}} \mathbb{E}_m[r^{\alpha,\beta}_m|\mathcal{F}_{m-1}] = \sum_{m=1}^{N_{ep}} \mathbb{E}_m[\mathrm{cost}_m(D_m) - \alpha \cdot \mathrm{cost}_m(D^*_m) - \beta|\mathcal{F}_{m-1}]. \tag{47}$$

Recall that, $\mathbb{E}_m[\mathrm{cost}_m(D_m)|\mathcal{F}_{m-1}] = \mathbb{E}_m[S_m(D_m)|\mathcal{F}_{m-1}] + \mathbb{E}_m[M_m(D_m)|\mathcal{F}_{m-1}]$. In the following sections we bound this sum of expected episodic costs using results from [19] and [18]. We begin by explicitly writing the episodic service costs in terms of the actions' distributions.

### E.1 Costs in Terms of Conditional Distribution

Recall in Algorithm 1 the sequence of decisions $D_m = \{x_{1,m}, \ldots, x_{H,m}\}$ is sampled from conditional optimal coupling distribution denoted as $\zeta_{h-1,h,m}(U_{h,m} = x|U_{h-1,m} = x_{h-1,m})$ as stated in Algorithm 1, Line 13. Here $(U_{h-1,m}, U_{h,m})$ is a joint random variable whose marginal distributions are $l(z_{h-1,m})$ and $l(z_{h,m})$, respectively, and are computed as stated in Algorithm 1, Line 11.

Hence, the expected service cost of the algorithm w.r.t. $\mathrm{lcb}_m(\cdot, e_{h,m})$ given $x_{h-1,m}$ becomes

$$\sum_{h=1}^H \mathbb{E}[\mathrm{lcb}_m(x_{h,m}, e_{h,m})|x_{h-1,m}, \mathcal{F}_{m-1}] = \sum_{h=1}^H \langle \mathrm{lcb}_m(\cdot, e_{h,m}), \zeta_{h-1,h,m}(\cdot|x_{h-1,m})\rangle.$$

Here the expectation $\mathbb{E}$ is w.r.t. the random variable $x_{h,m}$ whose probability distribution is $\zeta_{h-1,h,m}(U_{h,m} = x|U_{h-1,m} = x_{h-1,m})$ when conditioned on $x_{h-1,m}$. We note that the first

state of each episode is sampled from $l(z_{1,m})$ (as there is no randomness in the initial state $x_{0,m}$ [3]). Also, within any given episode, $\text{lcb}_m(\cdot,\cdot)$ in GP-MD is fixed and does not get updated. Hence, by taking the total expectation over the whole episode and by using the law of total expectation [4], we arrive at:

$$\mathbb{E}_m\Big[\sum_{h=1}^{H}\langle \text{lcb}_m(\cdot, e_{h,m}), \zeta_{h-1,h,m}(\cdot|x_{h-1,m})\rangle|\mathcal{F}_{m-1}\Big] = \sum_{h=1}^{H}\langle \text{lcb}_m(\cdot, e_{h,m}), l(z_{h,m})\rangle. \tag{48}$$

On the other hand, the actual service cost for episode $m$ is

$$S_m(D_m) = \sum_{h=1}^{H} f(x_{h,m}, e_{h,m}) \quad \text{where } \; x_{h,m} \sim \zeta_{h-1,h,m}(\cdot|x_{h-1,m}).$$

Hence, we can write

$$\mathbb{E}[f(x_{h,m}, e_{h,m})|x_{h-1,m}, \mathcal{F}_{m-1}] = \langle f(\cdot, e_{h,m}), \zeta_{h-1,h,m}(\cdot|x_{h-1,m})\rangle,$$

and

$$\sum_{h=1}^{H}\mathbb{E}[f(x_{h,m}, e_{h,m})|x_{h-1,m}, \mathcal{F}_{m-1}] = \sum_{h=1}^{H}\langle f(\cdot, e_{h,m}), \zeta_{h-1,h,m}(\cdot|x_{h-1,m})\rangle.$$

Moreover, conditioning on the event of Lemma 1, the cumulative cost above can be bounded in terms of its lower confidence bound as

$$\begin{aligned}\sum_{h=1}^{H}\langle f(\cdot, e_{h,m}), \zeta_{h-1,h,m}(\cdot|x_{h-1,m})\rangle \leq &\sum_{h=1}^{H}\langle \text{lcb}_m(\cdot, e_{h,m}), \zeta_{h-1,h,m}(\cdot|x_{h-1,m})\rangle \\ &+ \sum_{h=1}^{H}\langle 2\beta_m\sigma_m(\cdot, e_{h,m}), \zeta_{h-1,h,m}(\cdot|x_{h-1,m})\rangle.\end{aligned} \tag{49}$$

Also, by using the similar argument that led to Eq. (48), we have,

$$\mathbb{E}_m[S_m(D_m)|\mathcal{F}_{m-1}] = \mathbb{E}_m\Big[\sum_{h=1}^{H}\langle f(\cdot, e_{h,m}), \zeta_{h-1,h,m}(\cdot|x_{h-1,m})\rangle|\mathcal{F}_{m-1}\Big] = \sum_{h=1}^{H}\langle f(\cdot, e_{h,m}), l(z_{h,m})\rangle. \tag{50}$$

Then, combining Eq. (48), Eq. (49) and Eq. (50) we obtain

$$\begin{aligned}\sum_{h=1}^{H}\langle f(\cdot, e_{h,m}), l(z_{h,m})\rangle \leq &\sum_{h=1}^{H}\langle \text{lcb}_m(\cdot, e_{h,m}), l(z_{h,m})\rangle \\ &+ \mathbb{E}_m[\sum_{h=1}^{H}\langle 2\beta_m\sigma_m(\cdot, e_{h,m}), \zeta_{h-1,h,m}(\cdot|x_{h-1,m})\rangle|\mathcal{F}_{m-1}].\end{aligned} \tag{51}$$

By summing over all episodes (note that Lemma 1 holds uniformly over all episodes), we arrive at

$$\begin{aligned}\sum_{m=1}^{N_{ep}}\sum_{h=1}^{H}\langle f(\cdot, e_{h,m}), l(z_{h,m})\rangle \leq &\sum_{m=1}^{N_{ep}}\sum_{h=1}^{H}\langle \text{lcb}_m(\cdot, e_{h,m}), l(z_{h,m})\rangle \\ &+ \sum_{m=1}^{N_{ep}}\mathbb{E}_m[\sum_{h=1}^{H}\langle 2\beta_m\sigma_m(\cdot, e_{h,m}), \zeta_{h-1,h,m}(\cdot|x_{h-1,m})\rangle|\mathcal{F}_{m-1}].\end{aligned} \tag{52}$$

[3] The optimal coupling conditional distribution minimizing Eq. (25) for $h = 1$ will be trivially satisfied by $l(z_{1,m})$ as the random variable $U_{0,m}$ is fixed at $x_{0,m}$

[4] $\mathbb{E}_{x_{1,m},x_{2,m}}[\text{lcb}_m(x_{2,m}, e_{2,m})] = \sum_{y\in\mathcal{X}}\sum_{x\in\mathcal{X}}\text{lcb}_m(x, e_{h,m})\zeta_{1,2,m}(x|y)l(z_{1,m})(y)] = \sum_{x\in\mathcal{X}}\text{lcb}_m(x, e_{h,m})\sum_{y\in\mathcal{X}}\zeta_{1,2,m}(x, y) = \sum_{x\in\mathcal{X}}\text{lcb}_m(x, e_{h,m})l(z_{2,m})(x)$ (can be similarly proved for any $h$ using an inductive argument)

Now we define the offline optimal sequence for the cost sequence provided to Algorithm 1 in episode $m$ $\{\text{lcb}_m(\cdot, e_{1,m}), \ldots, \text{lcb}_m(\cdot, e_{H,m})\}$ as $\{bx^*_{1,m}, \ldots, bx^*_{H,m}\}$ which will be useful in the analysis to bound $\sum_{m=1}^{N_{ep}} \sum_{h=1}^{H} \langle \text{lcb}_m(\cdot, e_{h,m}), l(z_{h,m})\rangle$. Note that this offline optimal sequence is different from $D^*_m$ as it is calculated for the cost sequence $\{\text{lcb}_m(\cdot, e_{1,m}), \ldots, \text{lcb}_m(\cdot, e_{H,m})\}$ and not $\{f(\cdot, e_{1,m}), \ldots, f(\cdot, e_{H,m})\}$.

In Algorithm 1, we use the mirror descent approach as proposed in [19] with input as the cost sequence $\{\text{lcb}_m(\cdot, e_{1,m}), \ldots, \text{lcb}_m(\cdot, e_{H,m})\}$ to output the actions $D_m$. Since Algorithm 1 uses $\text{lcb}_m(\cdot, e_{h,m})$ rather than $f(\cdot, e_{h,m})$ the guarantees provided in [19] for action sequence $D_m$ will hold only w.r.t. $\{bx^*_{1,m}, \ldots, bx^*_{H,m}\}$ (the offline optimal sequence w.r.t. $\{\text{lcb}_m(\cdot, e_{1,m}), \ldots, \text{lcb}_m(\cdot, e_{H,m})\}$). Hence we can invoke Coester and Lee [19, Corollary 4], for the probability vector sequence $\{l(z_{1,m}), \ldots, l(z_{H,m})\}$ to guarantee that the sequence is 1-competitive in service costs (w.r.t. $\{\text{lcb}_m(\cdot, e_{1,m}), \ldots, \text{lcb}_m(\cdot, e_{H,m})\}$) and $\mathcal{O}((\log n)^2)-$competitive for movement costs w.r.t. the offline optimal sequence $\{bx^*_{1,m}, \ldots, bx^*_{H,m}\}$. Using the definition of refined competitive ratio guarantees Eq. (18) and Eq. (19) we have,

$$\sum_{h=1}^{H} \langle \text{lcb}_m(\cdot, e_{h,m}), l(z_{h,m})\rangle \leq \mathcal{O}(1) + \sum_{h=1}^{H} \Big( \text{lcb}_m(bx^*_{h,m}, e_{h,m}) + d(bx^*_{h,m}, bx^*_{h-1,m}) \Big), \tag{53}$$

$$\mathbb{E}_m[M_m(D_m)|\mathcal{F}_{m-1}] \leq \mathcal{O}(1) + \mathcal{O}((\log n)^2) \sum_{h=1}^{H} \Big( \text{lcb}_m(bx^*_{h,m}, e_{h,m}) + d(bx^*_{h,m}, bx^*_{h-1,m}) \Big). \tag{54}$$

Focusing on $\sum_{m=1}^{N_{ep}} \sum_{h=1}^{H} \langle \text{lcb}_m(\cdot, e_{h,m}), l(z_{h,m})\rangle$ in Eq. (52) and by using Eq. (53), we have

$$\sum_{m=1}^{N_{ep}} \sum_{h=1}^{H} \langle \text{lcb}_m(\cdot, e_{h,m}), l(z_{h,m})\rangle \leq \sum_{m=1}^{N_{ep}} \Big( \mathcal{O}(1) + \sum_{h=1}^{H} \big( \text{lcb}_m(bx^*_{h,m}, e_{h,m}) + d(bx^*_{h,m}, bx^*_{h-1,m}) \big) \Big). \tag{55}$$

Since $D^*_m = \{x^*_{1,m}, \ldots, x^*_{H,m}\}$ is not optimal w.r.t. $\text{lcb}_m(\cdot, e_{h,m})$, it will incur more cost than $\{bx^*_{1,m}, \ldots, bx^*_{H,m}\}$ w.r.t. $\text{lcb}_m(\cdot, e_{h,m})$, i.e,

$$\sum_{h=1}^{H} \Big( \text{lcb}_m(bx^*_{h,m}, e_{h,m}) + d(bx^*_{h,m}, bx^*_{h-1,m}) \Big) \leq \sum_{h=1}^{H} \Big( \text{lcb}_m(x^*_{h,m}, e_{h,m}) + d(x^*_{h,m}, x^*_{h-1,m}) \Big). \tag{56}$$

Hence, by plugging Eq. (56) in Eq. (55) we have ,

$$\sum_{m=1}^{N_{ep}} \sum_{h=1}^{H} \langle \text{lcb}_m(\cdot, e_{h,m}), l(z_{h,m})\rangle \leq \sum_{m=1}^{N_{ep}} \Big( \mathcal{O}(1) + \sum_{h=1}^{H} \big( \text{lcb}_m(x^*_{h,m}, e_{h,m}) + d(x^*_{h,m}, x^*_{h-1,m}) \big) \Big) \tag{57}$$

$$\leq \sum_{m=1}^{N_{ep}} \Big( \mathcal{O}(1) + \sum_{h=1}^{H} \big( f(x^*_{h,m}, e_{h,m}) + d(x^*_{h,m}, x^*_{h-1,m}) \big) \Big), \tag{58}$$

where the last equation Eq. (58) holds because we have conditioned on the event that confidence bounds hold true simultaneously for all episodes.

We can now use the above results to bound the cumulative service cost as

$$\sum_{m=1}^{N_{ep}} \mathbb{E}_m[S_m|\mathcal{F}_{m-1}] \overset{\text{(i)}}{=} \sum_{m=1}^{N_{ep}} \sum_{h=1}^{H} \langle f(\cdot, e_{h,m}), l(z_{h,m})\rangle \tag{59}$$

$$\begin{aligned} &\overset{\text{(ii)}}{\leq} \sum_{m=1}^{N_{ep}} \sum_{h=1}^{H} \langle \text{lcb}_m(\cdot, e_{h,m}), l(z_{h,m})\rangle \\ &+ \sum_{m=1}^{N_{ep}} \mathbb{E}_m[\sum_{h=1}^{H} \langle 2\beta_m \sigma_m(\cdot, e_{h,m}), \zeta_{h-1,h,m}(\cdot|x_{h-1,m})\rangle|\mathcal{F}_{m-1}]. \end{aligned} \tag{60}$$

$$\begin{aligned} &\overset{\text{(iii)}}{\leq} \sum_{m=1}^{N_{ep}} \Big(\mathcal{O}(1) + \sum_{h=1}^{H} \big(f(x^*_{h,m}, e_{h,m}) + d(x^*_{h,m}, x^*_{h-1,m})\big)\Big) \\ &+ \sum_{m=1}^{N_{ep}} \mathbb{E}_m[\sum_{h=1}^{H} \langle 2\beta_m \sigma_m(\cdot, e_{h,m}), \zeta_{h-1,h,m}(\cdot|x_{h-1,m})\rangle|\mathcal{F}_{m-1}]. \end{aligned} \tag{61}$$

Here (i) follows from Eq. (50), (ii) from Eq. (52) and (iii) from Eq. (58). Finally, we focus on bounding movement cost by using Eq. (54) and Eq. (58):

$$\sum_{m=1}^{N_{ep}} \mathbb{E}_m\Big[M_m(D_m)|\mathcal{F}_{m-1}\Big] = \sum_{m=1}^{N_{ep}} \mathbb{E}_m\Big[\sum_{h=1}^{H} d(x_{h,m}, x_{h-1,m})|\mathcal{F}_{m-1}\Big] \tag{62}$$

$$\leq \sum_{m=1}^{N_{ep}} \Big(\mathcal{O}(1) + \mathcal{O}((\log n)^2) \sum_{h=1}^{H} \big(\text{lcb}_m(bx^*_{h,m}, e_{h,m}) + d(bx^*_{h,m}, bx^*_{h-1,m})\big)\Big) \tag{63}$$

$$\leq \sum_{m=1}^{N_{ep}} \Big(\mathcal{O}(1) + \mathcal{O}((\log n)^2) \sum_{h=1}^{H} \big(f(x^*_{h,m}, e_{h,m}) + d(x^*_{h,m}, x^*_{h-1,m})\big)\Big). \tag{64}$$

### E.2 Bounding Learning Error

As a last step, we focus on bounding the second term in Eq. (52) that we refer to as the *learning error*, i.e., $\sum_{m=1}^{N_{ep}} \mathbb{E}_m\Big[\sum_{h=1}^{H} \langle 2\beta_m \sigma_m(\cdot, e_{h,m}), \zeta_{h-1,h,m}(\cdot|x_{h-1,m})\rangle|\mathcal{F}_{m-1}\Big]$. After that, we can finally bound the cumulative regret using the previously obtained bounds on service and movement costs (Eq. (61)and Eq. (64), respectively).

Consider the stochastic process,

$$\Delta_m = \sum_{h=1}^{H} \sigma_m^2(x_{h,m}, e_{h,m}).$$

Here $\sigma_m(\cdot,\cdot)$ is constructed from the data up to $m-1$ episodes defined earlier as $\mathcal{F}_{m-1}$. Now, using a similar argument that led to Eq. (48) (fixed $\sigma_m(\cdot,\cdot)$ for any given episode and law of total expectation), the conditional mean of $\Delta_m$ given is $\mathcal{F}_{m-1}$

$$\mathbb{E}_m[\Delta_m|\mathcal{F}_{m-1}] = \sum_{h=1}^{H} \langle \sigma_m^2(\cdot, e_{h,m}), l(z_{h,m})\rangle.$$

Now in order to bound this sum of conditional means of posterior variance $\sum_{m=1}^{N_{ep}} \mathbb{E}_m[\Delta_m|\mathcal{F}_{m-1}]$, by observed posterior variance $\sum_{m=1}^{N_{ep}} \Delta_m$ we use [30, Lemma 3]. Note that $\sigma_m^2(x, e) \leq 1$ by our

assumption $k(\cdot,\cdot) \leq 1$ in Section 1 and the stochastic process $\Delta_m$ can be bounded as follows

$$\Delta_m = \sum_{h=1}^{H} \sigma_m^2(x_{h,m}, e_{h,m}) \leq \sum_{h=1}^{H}(1) \leq H.$$

Hence applying [30, Lemma 3] with probability at least $1-\delta$,

$$\sum_{m=1}^{N_{ep}} \mathbb{E}_m[\Delta_m|\mathcal{F}_{m-1}] \leq 2\Big(\sum_{m=1}^{N_{ep}}(\Delta_m)\Big) + 4H\log(1/\delta) + 8H\log(4H) + 1,$$

This implies,

$$\sum_{m=1}^{N_{ep}}\sum_{h=1}^{H}\langle\sigma_m^2(\cdot, e_{h,m}), l(z_{h,m})\rangle \leq 2\sum_{m=1}^{N_{ep}}\sum_{h=1}^{H}\sigma_m^2(x_{h,m}, e_{h,m}) + 4H\log(1/\delta) + 8H\log(4H) + 1. \tag{65}$$

Note that Eq. (65) cannot be directly bounded using bounds for sum of observed posterior variance as done in [50] and [17]. This is because $\sigma_m(\cdot,\cdot)$ is not updated continuously and is constant within any given episode $m$. Hence we first need to bound $\sum_{m=1}^{N_{ep}}\sum_{h=1}^{H}\sigma_m^2(x_{h,m}, e_{h,m})$ by $\sum_{m=1}^{N_{ep}}\sum_{h=1}^{H}\sigma_{h,m}^2(x_{h,m}, e_{h,m})$ where $\sigma_{h,m}(\cdot,\cdot)$ is constructed based on all $\{(x_{h,m}, e_{h,m}, y_{h,m})\}$ observed up to the round $h$ in the $m$-th episode. Towards this end, we recall the following Lemma from Chowdhury and Gopalan [18].

**Lemma 2.** *Let $k : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$ be a symmetric positive semi-definite kernel such that it has bounded variance, i.e. $k(x,x) \leq 1$ for all $x \in \mathcal{X}$ and $f \sim GP_{\mathcal{X}}(0,k)$ be a sample from the associated Gaussian process over $\mathcal{X}$, then for all $s \geq 1$ and $x \in X$,*

$$\sigma_{s-1}^2(x) \leq (1+\lambda^{-1})\sigma_s^2(x), \tag{66}$$

*and*

$$\sum_{s=1}^{t}\sigma_{s-1}^2 \leq (1+2\lambda)\sum_{s=1}^{t}\frac{1}{2}\ln(1+\lambda^{-1}\sigma_{s-1}(x_s)) \leq 2\lambda\gamma_t(k,\mathcal{X}). \tag{67}$$

Following the proof technique of Chowdhury and Gopalan [18, Lemma-11] and using Eq. (66) and Eq. (67) to bound $\sum_{m=1}^{N_{ep}}\sum_{h=1}^{H}\sigma_m^2(x_{h,m}, e_{h,m})$, we have for $\lambda = H$,

$$\sum_{m=1}^{N_{ep}}\sum_{h=1}^{H}\sigma_m^2(x_{h,m}, e_{h,m}) \leq \sum_{m=1}^{N_{ep}}\Big(\sum_{h=1}^{H}(1+1/H)^{h-1}\sigma_{h,m}^2(x_{h,m}, e_{h,m})\Big) \tag{68}$$

$$\leq (1+1/H)^{H-1}\sum_{m=1}^{N_{ep}}\sum_{h=1}^{H}\sigma_{h,m}^2(x_{h,m}, e_{h,m}) \tag{69}$$

$$\leq (1+1/H)^{H-1}(1+2H)\sum_{m=1}^{N_{ep}}\sum_{h=1}^{H}\frac{1}{2}\ln\Big(1+\frac{1}{H}\sigma_{h,m}^2(x_{h,m}, e_{h,m})\Big) \tag{70}$$

$$\leq 2eH\gamma_{N_{ep}H}. \tag{71}$$

Here the last inequality is due to $(1+H^{-1})^H \leq e$ and $(1+H^{-1})^{-1}(2H+1) \leq 2H$.

Now, we are in position to focus on the learning error. It turns out that the learning error can be simplified by using similar arguments as in Eq. (48), i.e.,

$$\sum_{m=1}^{N_{ep}}\mathbb{E}_m\Big[\sum_{h=1}^{H}\langle 2\beta_m\sigma_m(\cdot, e_{h,m}), \zeta_{h-1,h,m}(\cdot|x_{h-1,m})\rangle\Big|\mathcal{F}_{m-1}\Big] = \sum_{m=1}^{N_{ep}}\sum_{h=1}^{H}2\beta_m\langle\sigma_m(\cdot, e_{h,m}), l(z_{h,m})\rangle. \tag{72}$$

As $\beta_m$ is a non-decreasing sequence as stated in Lemma 1, we have

$$\sum_{m=1}^{N_{ep}}\sum_{h=1}^{H} 2\beta_m\langle\sigma_m(\cdot,e_{h,m}),l(z_{h,m})\rangle \le 2\beta_{N_{ep}}\sum_{m=1}^{N_{ep}}\sum_{h=1}^{H}\langle\sigma_m(\cdot,e_{h,m}),l(z_{h,m})\rangle \tag{73}$$

$$\le 2\beta_{N_{ep}}\sqrt{N_{ep}H\sum_{m=1}^{N_{ep}}\sum_{h=1}^{H}(\langle\sigma_m(\cdot,e_{h,m}),l(z_{h,m})\rangle)^2} \tag{74}$$

$$\le 2\beta_{N_{ep}}\sqrt{N_{ep}H\sum_{m=1}^{N_{ep}}\sum_{h=1}^{H}\langle\sigma_m^2(\cdot,e_{h,m}),l(z_{h,m})\rangle}. \tag{75}$$

We obtain Eq. (74) using Cauchy-Schwartz inequality and Eq. (75) using Jensen's inequality since $l(z_{h,m})$ is a probability distribution.

Finally, the learning error will be bounded by

$$\begin{aligned}
&\sum_{m=1}^{N_{ep}}\mathbb{E}_m\sum_{h=1}^{H}\langle 2\beta_m\sigma_m(\cdot,e_{h,m}),\zeta_{h-1,h,m}(\cdot|x_{h-1,m})\rangle|\mathcal{F}_{m-1}] \\
&\qquad\overset{\text{(i)}}{\le} 2\beta_{N_{ep}}\sqrt{N_{ep}H\sum_{m=1}^{N_{ep}}\sum_{h=1}^{H}\langle\sigma_m^2(\cdot,e_{h,m}),l(z_{h,m})\rangle} \qquad (76)\\
&\overset{\text{(ii)}}{\le} 2\beta_{N_{ep}}\sqrt{2N_{ep}H\sum_{m=1}^{N_{ep}}\sum_{h=1}^{H}\sigma_m^2(x_{h,m},e_{h,m})+4H\log(1/\delta)+8H\log(4H)+1} \qquad (77)\\
&\overset{\text{(iii)}}{\le} \mathcal{O}\Big(\beta_{N_{ep}}\sqrt{N_{ep}H(2eH\gamma_{N_{ep}H}+4H\log(1/\delta)+8H\log(4H)+1)}\Big). \qquad (78)
\end{aligned}$$

Here (i) follows from Eq. (75), (ii) from Eq. (65) and (iii) from Eq. (71).

### E.3 Bounding the regret

To bound the cumulative regret, recall our initial goal in Eq. (47) to bound

$$\sum_{m=1}^{N_{ep}}\mathbb{E}_m[\text{cost}_m(D_m)|\mathcal{F}_{m-1}] = \sum_{m=1}^{N_{ep}}\mathbb{E}_m[S_m(D_m)+M_m(D_m)|\mathcal{F}_{m-1}].$$

By using Eq. (78) in Eq. (61), we bound the service costs $\sum_{m=1}^{N_{ep}}\mathbb{E}_m[S_m(D_m)|\mathcal{F}_{m-1}]$ as

$$\begin{aligned}
\sum_{m=1}^{N_{ep}}\mathbb{E}_m[S_m(D_m)|\mathcal{F}_{m-1}] \le &\sum_{m=1}^{N_{ep}}\Big(\mathcal{O}(1)+\sum_{h=1}^{H}\big(f(x_{h,m}^*,e_{h,m})+d(x_{h,m}^*,x_{h-1,m}^*)\big)\Big) \\
&+\mathcal{O}\Big(\beta_{N_{ep}}\sqrt{N_{ep}H(2eH\gamma_{N_{ep}H}+4H\log(1/\delta)+8H\log(4H)+1)}\Big).
\end{aligned} \tag{79}$$

Also, Eq. (64) bounds movement costs as follows:

$$\sum_{m=1}^{N_{ep}}\mathbb{E}_m[M_m(D_m)|\mathcal{F}_{m-1}] \le \sum_{m=1}^{N_{ep}}\Big(\mathcal{O}(1)+\mathcal{O}((\log n)^2)\sum_{h=1}^{H}\big(f(x_{h,m}^*,e_{h,m})+d(x_{h,m}^*,x_{h-1,m}^*)\big)\Big).$$

Combining this we obtain,

$$\sum_{m=1}^{N_{ep}} \mathbb{E}_m[\text{cost}_m(D_m)|\mathcal{F}_{m-1}] = \sum_{m=1}^{N_{ep}} \mathbb{E}_m[S_m(D_m) + M_m(D_m)|\mathcal{F}_{m-1}] \tag{80}$$

$$\leq \sum_{m=1}^{N_{ep}} \Big(\mathcal{O}(1) + \mathcal{O}((\log n)^2) \sum_{h=1}^{H} \big(f(x^*_{h,m}, e_{h,m}) + d(x^*_{h,m}, x^*_{h-1,m})\big)\Big) + \mathcal{O}\Big(\beta_{N_{ep}} \big(N_{ep} H (2eH\gamma_{N_{ep}H} + 4H\log(1/\delta) + 8H\log(4H))\big)^{\frac{1}{2}}\Big) \tag{81}$$

$$\leq \sum_{m=1}^{N_{ep}} \Big(\mathcal{O}(1) + \mathcal{O}((\log n)^2) \cdot \text{cost}_m(D^*_m)\Big) + \mathcal{O}\Big(\beta_{N_{ep}} \big(N_{ep} H (2eH\gamma_{N_{ep}H} + 4H\log(1/\delta) + 8H\log(4H))\big)^{\frac{1}{2}}\Big). \tag{82}$$

Thus, by setting set $\alpha = \mathcal{O}((\log n)^2)$ and $\beta = \mathcal{O}(1)$ the expected regret $\sum_{m=1}^{N_{ep}} \mathbb{E}_m[R^{\alpha,\beta}_m|\mathcal{F}_{m-1}]$ from Eq. (46) can be bounded using Eq. (82) as

$$\sum_{m=1}^{N_{ep}} \mathbb{E}_m[r^{\alpha,\beta}_m|\mathcal{F}_{m-1}] = \sum_{m=1}^{N_{ep}} \mathbb{E}_m[\text{cost}_m(D_m) - \mathcal{O}((\log n)^2) \cdot \text{cost}_m(D^*_m) - \mathcal{O}(1)|\mathcal{F}_{m-1}] \tag{83}$$

$$\leq \mathcal{O}\Big(\beta_{N_{ep}} \big(N_{ep} H (2eH\gamma_{N_{ep}H} + 4H\log(1/\delta) + 8H\log(4H))\big)^{\frac{1}{2}}\Big). \tag{84}$$

Now, for the final step of the regret guarantees is to ensure that Eq. (84) holds with probability at least $1-\delta$. For this, we take the union bound over the events such that Lemma 1, Eq. (65) and Eq. (46) hold. This effectively replaces $\delta$ by $\delta/3$ in each of the statements. Hence we get the required regret guarantee with probability at least $1-\delta$ as stated in Thm. 1:

$$R^{\alpha,\beta}_{N_{ep}} \leq \mathcal{O}\Big(\beta_{N_{ep}} \big(H^2 N_{ep} \gamma_{HN_{ep}} + H\log(\tfrac{H}{\delta})\big)^{\frac{1}{2}} + H(B+\psi)\log\big(\tfrac{N_{ep}\log(N_{ep})}{\delta}\big)\Big). \tag{85}$$

# F Experimental Details

In this section, we explain how we arrived at the energy generation formula and objective function based on the discrete-time power generation formula (Eqs. (10), (11)) from [7]:

$$P(x,t) = c_1(\min\{V_w(x,t), V_r\})^3 - c_2 V_w^2(x,t) - c_3 V_r^2 \frac{|x-x'|}{\Delta t'}. \tag{86}$$

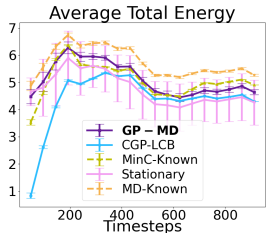
The first two terms in Eq. (86) determine the instantaneous power generated at a particular altitude. As we update altitudes only with a time gap of $\Delta t = 60$ min, the energy generated at this altitude for $\Delta t$ time would be $E_S(x,t) = \big(c_1(\min\{V_w(x,t), V_r\})^3 - c_2 V_w^2(x,t)\big)\Delta t$. Here $x$ denotes the altitude and corresponds to our action space and $t$ denotes to the time of day and corresponds to our contexts. Next, $x'$ denotes the altitude at time $t-1$, and $\Delta t'$ is the time taken to change altitude. After time interval $\Delta t'$, the altitude is fixed for the rest of $\Delta t$ and the third in Eq. (86) becomes zero. Also, $\Delta t'$ may or may not be equal to $\Delta t$ (usually, it is lower). But the energy lost in changing altitude will be $E_M(x,x') = c_3 V_r^2 \frac{|x-x'|}{\Delta t'} * \Delta t' = c_3 V_r^2 |x-x'|$. As we update $x$ only in intervals of $\Delta t$, the energy lost for movement for every $\Delta t$ time would be $E_M(x,x')$.

### F.1 Plots from Other Locations

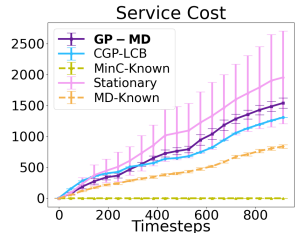
We present the remaining plots corresponding to the AWE altitude optimization task considered in Section 4.1:

- Fig. 10: First, we show additional plots that correspond to Loc. 2 (Lat. $= 53$, Long. $= -4$) whose total energy generated for varying $\rho$ was shown in Fig. 3f.
- Fig. 11: Next, we consider another Loc. 3 (Lat. $= 47$, Long. $= 6$; we used $\text{outputscale} = 12.94$).

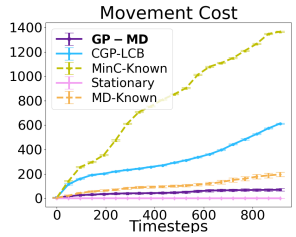
- Fig. 12: Lastly, we show the results that correspond to Loc. 1 (Lat. $= 53$, Long. $= -10$), but we use data from a different range (October-November 2016; again, we used $\mathrm{outputscale} = 12.94$).




(a) Total Energy Generated for $\rho = 4$

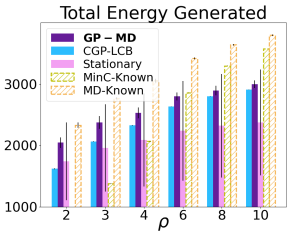



(b) Service Cost for $\rho = 4$

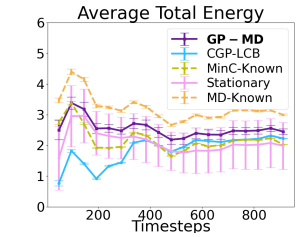



(c) Movement Cost for $\rho = 4$

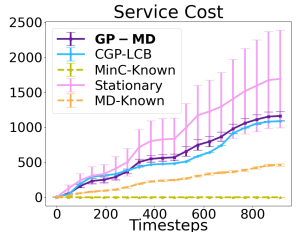
Figure 10: AWE altitude optimization task; Fig. 10a: The average total generated energy over 960 hours based on the wind data at a single location (Latitude$= 53$ and Longitude$= -4$) for $\rho = 4$. Figs. 10b and 10c: The service and movement costs for $\rho = 4$.




(a) Total Energy Generated for varying $\rho$

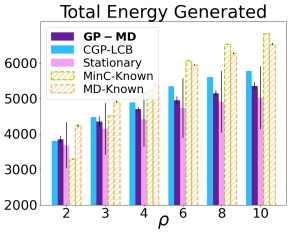



(b) Total Energy Generated for $\rho = 4$

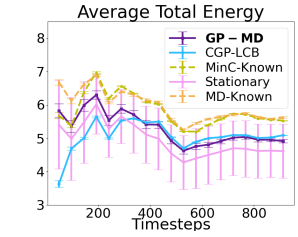



(c) Service Cost for $\rho = 4$

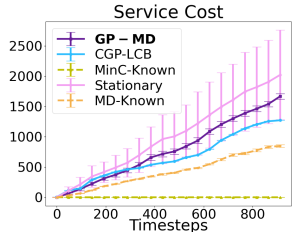
Figure 11: AWE altitude optimization task; Fig. 11a: Total energy generated for 960 hours based on the wind data at a single location (Latitude$= 47$ and Longitude$= 6$). GP-MD outperforms previously used CGP-LCB (that optimizes for service costs only) for a range of $\rho$ values that favor the service against the movement cost. Fig. 11b: The average total generated energy over 960 hours. Fig. 11c: The service cost for $\rho = 4$.




(a) Total Energy Generated for varying $\rho$




(b) Total Energy Generated for $\rho = 4$




(c) Service Cost for $\rho = 4$

Figure 12: AWE altitude optimization task; Fig. 12a: Total energy generated for 960 hours based on the wind data (different months from Fig. 2a) at a single location (Latitude$= 53$ and Longitude$= -10$). GP-MD outperforms previously used CGP-LCB (that optimizes for service costs only) only for relatively low $\rho$ values that favor the service against the movement cost. Fig. 12b: The average total generated energy over 960 hours. Fig. 12c: The service cost for $\rho = 4$.