Efficient Online Clustering with Moving Costs

Anonymous Author(s)
Affiliation
Address
email

Abstract

In this work we consider an online learning problem, called *Online k-Clustering with Moving Costs*, at which a *learner* maintains a set of k facilities over T rounds so as to minimize the connection cost of an adversarially selected sequence of clients. The *learner* is informed on the positions of the clients at each round t only after its facility-selection and can use this information to update its decision in the next round. However, updating the facility positions comes with an additional moving cost based on the moving distance of the facilities. We present the first $\mathcal{O}(\log n)$-regret polynomial-time online learning algorithm guaranteeing that the overall cost (connection + moving) is at most $\mathcal{O}(\log n)$ times the time-averaged connection cost of the *best fixed solution*. Our work improves on the recent result of Fotakis et al. [30] establishing $\mathcal{O}(k)$-regret guarantees *only* on the connection cost.

1 Introduction

Due to their various applications in diverse fields (e.g. machine learning, operational research, data science etc.), *clustering problems* have been extensively studied. In the well-studied k-median problem, given a set of clients, k facilities should be placed on a metric with the objective to minimize the sum of the distance of each client from its closest center [54, 14, 13, 66, 6, 43, 51, 64, 50, 15, 53, 3].

In many modern applications (e.g., epidemiology, social media, conference, etc.) the positions of the clients are not *static* but rather *evolve over time* [56, 55, 63, 58, 23, 5]. For example the geographic distribution of the clients of an online store or the distribution of Covid-19 cases may drastically change from year to year or respectively from day to day [30]. In such settings it is desirable to update/change the positions of the facilities (e.g., compositions of warehouses or Covid test-units) so as to better serve the time-evolving trajectory of the clients.

The clients' positions may change in complex and unpredictable ways and thus an *a priori knowledge* on their trajectory is not always available. Motivated by this, a recent line of research studies clustering problems under the *online learning framework* by assuming that the sequence of clients' positions is *unknown* and *adversarially selected* [18, 28, 16, 30]. More precisely, a *learner* must place k facilities at each round $t \geq 1$ without knowing the positions of clients at round t which are revealed to the learner only after its facility-selection. The learner can use this information to update its decision in the next round; however, moving a facility comes with an additional moving cost that should be taken into account in the learner's updating decision, e.g. moving Covid-19 test-units comes with a cost [18, 28].

Building on this line of works, we consider the following online learning problem:

Problem 1 (*Online k-Clustering with Moving Costs*)**.** *Let $G(V, E, w)$ be a weighted graph with $|V| = n$ vertices and k facilities. At each round $t = 1, \ldots, T$:*

1. *The learner selects $F_t \subseteq V$, with $|F_t| = k$, at which facilities are placed.*
2. *The adversary selects the clients' positions, $R_t \subseteq V$.*

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

3. The learner learns the clients' positions R_t and suffers

$$cost = \sum_{j \in R_t} \underbrace{\min_{i \in F_t} d_G(j,i)}_{\text{connection cost of client } j} + \underbrace{\gamma \cdot M_G(F_{t-1}, F_t)}_{\text{moving cost of facilities}}$$

where $d_G(j,i)$ is the distance between vertices $i, j \in V$; $M_G(F_{t-1}, F_t)$ is the minimum overall distance required to move k facilities from F_{t-1} to F_t; and $\gamma \geq 0$ is the facility-weight.

An *online learning algorithm* for Problem 1 tries to minimize the overall (connection + moving) cost by placing k facilities at each round $t \geq 1$ based only on the previous positions of clients $R_1, \ldots, R_{t-1}$. To the best of our knowledge, Problem 1 was first introduced in [18][1]. If for any sequence of clients, the overall cost of the algorithm is at most α times the overall connection cost of the *optimal fixed placement of facilities* F^* then the algorithm is called α-regret, while in the special case of $\alpha = 1$ the algorithm is additionally called *no-regret*.

Problem 1 comes as a special case of the well-studied *Metrical Task System* by considering each of the possible $\binom{n}{k}$ facility placements as a different state. In their seminal work, [11] guarantee that the famous *Multiplicative Weights Update algorithm* (MWU) achieves $(1+\epsilon)$-regret in Problem 1 for any $\epsilon > 0$. Unfortunately, running the MWU algorithm for Problem 1 is not really an option since it requires $\mathcal{O}(n^k)$ time and space complexity. As a result, the following question naturally arises:

Q. *Can we achieve α-regret for Problem 1 with polynomial-time online learning algorithms?*

Answering the above question is a challenging task. Even in the very simple scenario of time-invariant clients, i.e. $R_t = R$ for all $t \geq 1$, an α-regret online learning algorithm must essentially compute an *α-approximate solution* of the k-median problem. Unfortunately the k-median problem cannot be approximated with ratio $\alpha < 1 + 2/e \simeq 1.71$ (unless NP $\subseteq$ DTIME$[n^{\log \log n}]$ [42]) which excludes the existence of an $(1+2/e)$-regret polynomial-time online learning algorithm for Problem 1. Despite the fact that many $\mathcal{O}(1)$-approximation algorithms have been proposed for the k-median problem (the best current ratio is $1+\sqrt{3}$ [53]), these algorithms crucially rely on the (offline) knowledge of the whole sequence of clients and most importantly are not designed to handle the moving cost of the facilities [54, 14, 13, 66, 6, 43, 51, 64, 50, 15, 53, 3].

In their recent work, Fotakis et al. [30] propose an $\mathcal{O}(k)$-regret polynomial-time online learning algorithm for Problem 1 *without* moving costs (i.e. the special case of $\gamma = 0$). Their approach is based on designing a *no-regret* polynomial-time algorithm for a *fractional relaxation* of Problem 1 and then using an *online client-oblivious* rounding scheme in order to convert a fractional solution to an integral one. Their analysis is based on the fact that the connection cost of *any possible client* is at most $\mathcal{O}(k)$ times its fractional connection cost. However in order to establish the latter guarantee their rounding scheme performs abrupt changes on the facilities leading to huge moving cost.

Our Contribution and Techniques. In this work, we provide a positive answer to question (**Q**), by designing the first polynomial-time online learning algorithm for Online k-Clustering with Moving Costs that achieves $\mathcal{O}(\log n)$-regret for any $\gamma \geq 0$. The cornerstone idea of our work was to realize that $\mathcal{O}(1)$-regret can be established with a polynomial-time online learning algorithm in the special case of G being a Hierarchical Separation Tree (HST). Then, by using the standard metric embedding result of [25], we can easily convert such an algorithm to an $\mathcal{O}(\log n)$-regret algorithm for general graphs. Our approach for HSTs consists of two main technical steps:

1. We introduce a fractional relaxation of Problem 1 for HSTs. We then consider a specific regularizer on the fractional facility placements, called *Dilated Entropic Regularizer* [26], that takes into account the specific structure of the HST. Our first technical contribution is to establish that the famous *Follow the Leader algorithm* [34] with dilated entropic regularization admits $\mathcal{O}(1)$-regret for any $\gamma \geq 0$.
2. Our second technical contribution is the design of a novel *online client-oblivious* rounding scheme, called $\mathrm{Cut\&Round}$, that converts a fractional solution for HSTs into an integral one. By exploiting the specific HST structure we establish that $\mathrm{Cut\&Round}$, despite not

[1] In [18], an easier version of Problem 1 with 1-*lookahead* is considered, meaning that the learner learns the positions of the clients R_t before selecting F_t. Moreover, G is considered to be the line graph and $\gamma = 1$.

knowing the clients' positions R_t, simultaneously guarantees that (i) the connection cost of each client $j \in R_t$ is upper bounded by its fractional connection cost, and (ii) the expected moving cost of the facilities is at most $\mathcal{O}(1)$ times the fractional moving cost.

Experimental Evaluation. In Section F of the Appendix we experimentally compare our algorithm with the algorithm of Fotakis et al. [30]. Our experiments verify that our algorithm is robust to increases of the facility weight γ while the algorithm of [30] presents a significant cost increase. We additionally experimentally evaluate our algorithm in the MNIST and CIFAR10 datasets. Our experimental evaluations suggest that the $\mathcal{O}(\log n)$-regret bound is a pessimistic upper bound and that in practise our algorithm performs significantly better.

Related Work. As already mentioned, our work most closely relates with the work of Fotakis et al. [30] that provides an $\mathcal{O}(k)$-regret algorithm running in polynomial-time for $\gamma = 0$. [16] also consider Problem 1 for $\gamma = 0$ with the difference that the connection cost of clients is captured through the k-means objective i.e. the sum of the squared distances. They provide an $(1+\epsilon)$-regret algorithm with $\mathcal{O}\left((k^2/\epsilon^2)^{2k}\right)$ time-complexity that is still exponential in k. [18, 28] study the special case of Problem 1 in which G is the line graph and $\gamma = 1$ while assuming 1-*lookahead* on the request R_t. For $k = 1$, [18] provide an $(1+\epsilon)$-competitive online algorithm meaning that its cost is at most $(1+\epsilon)$ times the cost of the *optimal dynamic solution* and directly implies $(1+\epsilon)$-regret. [28] extended the previous result by providing a 63-competitive algorithm for $k = 2$ on line graphs. Our work also relates with the works of [23] and [4] that study offline approximation algorithms for clustering problems with *time-evolving metrics*. Finally our work is closely related with the research line of online learning in combinatorial domains and other settings of online clustering. Due to space limitations, we resume this discussion in Section A of the Appendix.

2 Preliminaries and Our Results

Let $G(V, E, w)$ be a weighted undirected graph where V denotes the set of vertices and E the set of edges among them. The weight w_e of an edge $e = (i, j) \in E$ denotes the cost of traversing e. Without loss, we assume that $w_e \in \mathbb{N}$ and $w_e \geq 1$ for all edges $e \in E$. The *distance* between vertices $i, j \in V$ is denoted with $d_G(i, j)$ and equals the cost of the minimum cost path from $i \in V$ to $j \in V$. We use $n := |V|$ to denote the cardinality of G and $D_G := \max_{i,j \in V} d_G(i, j)$ to denote its diameter.

Given a placement of facilities $F \subseteq V$, with $|F| = k$, a client placed at vertex $j \in V$ connects to the *closest open facility* $i \in F$. This is formally captured in Definition 1.

Definition 1. *The connection cost of a set of clients $R \subseteq V$ under the facility-placement $F \subseteq V$ with $|F| = k$ equals*

$$C_R(F) := \sum_{j \in R} \min_{i \in F} d_G(j, i)$$

Next, consider any pair of facility-placements $F, F' \subseteq V$ such that $|F| = |F'| = k$. The moving distance between F and F' is the minimum overall distance needed to transfer the k facilities from F to F', formally defined in Definition 2.

Definition 2. *Fix any facility-placements $F, F' \subseteq V$ where $|F| = |F'| = k$. Let Σ be the set of all possible matchings from F to F', i.e. each $\sigma \in \Sigma$ is a one-to-one mapping $\sigma : F \mapsto F'$ with $\sigma(i) \in F'$ denoting the mapping of facility $i \in F$. The moving cost between F and F' equals*

$$M_G(F, F') := \min_{\sigma \in \Sigma} \sum_{i \in F} d_G(i, \sigma(i))$$

At each round $t \geq 1$, an online learning algorithm $\mathcal{A}$ for Problem 1 takes as input all the *previous* positions of the clients $R_1, \ldots, R_{t-1} \subseteq V$ and outputs a facility-placement $F_t := \mathcal{A}(R_1, \ldots, R_{t-1})$ such that $F_t \subseteq V$ and $|F_t| = k$. The performance of an online learning algorithm is measured by the notion of *regret*, which we formally introduce in Definition 3.

Definition 3. *An online learning algorithm $\mathcal{A}$ for Problem 1 is called α-regret with additive regret β if and only if for any sequence of clients $R_1, \ldots, R_T \subseteq V$,*

$$\mathbb{E}\left[\sum_{t=1}^{T} C_{R_t}(F_t) + \gamma \cdot \sum_{t=2}^{T} M_G(F_{t-1}, F_t)\right] \leq \alpha \cdot \min_{|F^*|=k} \sum_{t=1}^{T} C_{R_t}(F^*) + \beta \cdot \sqrt{T}$$

where $F_t = \mathcal{A}(R_1, \ldots, R_{t-1})$ and α, β are constants independent of T.

An online learning algorithm $\mathcal{A}$ selects the positions of the k facilities at each round $t \geq 1$ solely based on the positions of the clients in the previous rounds, $R_1, \ldots, R_{t-1}$. If $\mathcal{A}$ is α-regret then Definition 3 implies that its time-averaged overall cost (connection + moving cost) is at most α times the time-averaged cost of the *optimal static solution!* [2] Furthermore, the dependency on $\sqrt{T}$ is known to be optimal [11] and β is typically only required to be polynomially bounded by the size of the input, as for $T \to \infty$ the corresponding term in the time-averaged cost vanishes.

As already mentioned, the seminal work of [11] implies the existence of an $(1+\epsilon)$-regret algorithm for Problem 1, however, this algorithm requires $\mathcal{O}(n^k)$ time and space complexity. Prior to this work, the only polynomial time online learning algorithm for Problem 1 was due to Fotakis et al. [30], for the special case of $\gamma = 0$. Specifically, in their work the authors design an online learning algorithm with the following guarantee:

Theorem (Fotakis et al. [30]). *There exists a randomized online learning algorithm for Problem 1 that runs in polynomial time (w.r.t. T, n and $\log D_G$) such that*

$$\mathbb{E}\left[\sum_{t=1}^{T} C_{R_t}(F_t)\right] \leq \mathcal{O}(k) \cdot \min_{|F^*|=k} \sum_{t=1}^{T} C_{R_t}(F^*) + \mathcal{O}(k \cdot n \cdot \sqrt{\log n} \cdot D_G) \cdot \sqrt{T}$$

Clearly, the algorithm of [30] has not been designed to account for charging the moving of facilities, as indicated by the absence of the moving cost in the above regret guarantee. The main contribution of this work is to obtain (for the first time) regret guarantees that also account for the moving cost.

Theorem 1. *There exists a randomized online learning algorithm for Problem 1 (Algorithm 2) that runs in polynomial time (w.r.t. T, n and $\log D_G$) and admits the following regret guarantee:*

$$\mathbb{E}\left[\sum_{t=1}^{T} C_{R_t}(F_t) + \gamma \cdot \sum_{t=2}^{T} M_G(F_{t-1}, F_t)\right] \leq \mathcal{O}(\log n) \cdot \min_{|F^*|=k} \sum_{t=1}^{T} C_{R_t}(F^*) + \beta \cdot \sqrt{T}$$

for $\beta = \mathcal{O}(k \cdot n^{3/2} \cdot D_G \cdot \max(\gamma, 1))$ and any $\gamma \geq 0$.

Remark 1. *We remark that while our additive regret β is larger than the corresponding term in [30] by a factor of $o(\sqrt{n})$, our results apply to any $\gamma \geq 0$ while the algorithm of [30] can generally suffer unbounded moving cost for $\gamma \to \infty$, as our experimental results verify.*

2.1 HSTs and Metric Embeddings

In this section we provide some preliminary introduction to Hierarchical Separation Trees (HSTs), as they consist a key technical tool towards proving Theorem 1. A *weighted tree* $\mathcal{T}(V, E, w)$ is a weighted graph with no cycles. Equivalently, for any pair of vertices $i, j \in V$ there exists a unique path that connects them. In Definition 4, we establish some basic notation for tree graphs.

Definition 4. *Fix any tree $\mathcal{T}(V, E, w)$. For every vertex $u \in V$, $\mathrm{cld}(u) \subseteq V$ denotes the set children vertices of u and $p(u)$ denotes its unique parent, i.e. $u \in \mathrm{cld}(p(u))$. The root $r \in V$ of $\mathcal{T}$ is the unique node with $p(r) = \varnothing$ and the set $L(\mathcal{T}) := \{u \in V : \mathrm{cld}(u) = \varnothing\}$ denotes the leaves of $\mathcal{T}$. We use $\mathrm{dpt}(u)$ to denote the depth of a vertex $u \in V$, i.e. the length of the (unique) path from the root r to u, and $h(\mathcal{T}) := \max_{u \in L(\mathcal{T})} \mathrm{dpt}(u)$ to denote the height of $\mathcal{T}$. We use $\mathrm{lev}(u) := h(\mathcal{T}) - \mathrm{dpt}(u)$ to denote the level of a vertex $u \in V$. Finally, $T(u) \subseteq V$ denotes the set of vertices on the sub-tree rooted at u, i.e. the set of vertices that are descendants of u.*

Next, we proceed to define a family of well-structured tree graphs that constitute one of the primary technical tools used in our analysis.

Definition 5. *A Hierarchical Separation Tree (HST) is a weighted tree $\mathcal{T}(V, E, w)$ such that (i) for any node u and any of its children $v \in cld(u)$, the edge $e = (u, v)$ admits weight $w_e = 2^{\mathrm{lev}(v)}$, and (ii) the tree is balanced, namely $lev(u) = 0$ for all leaves $u \in L(\mathcal{T})$.*

In their seminal works, [10] and later [24] showed that HSTs can approximately preserve the distances of any graph $G(V, E, w)$ within some logarithmic level of distortion.

[2] Specifically, the time-averaged overall cost of $\mathcal{A}$ approaches this upper bound with rate $\beta \cdot T^{-1/2}$.

Theorem 2. *For any graph $G(V, E, w)$ with $|V| = n$ and diameter D, there exists a polynomial-time randomized algorithm that given as input G produces an HST $\mathcal{T}$ with height $h(\mathcal{T}) \leq \lceil \log D \rceil$ s.t.*

1. *$L(\mathcal{T}) = V$, meaning that the leaves of $\mathcal{T}$ correspond to the vertices of G.*
2. *For any $u, v \in V$, $d_G(u, v) \leq d_{\mathcal{T}}(u, v)$ and $\mathbb{E}[d_{\mathcal{T}}(u, v)] \leq \mathcal{O}(\log n) \cdot d_G(u, v)$.*

Theorem 2 states that any weighted graph $G(V, E, w)$ can be embedded into an HST $\mathcal{T}$ with $\mathcal{O}(\log n)$-distortion. This means that the distance $d_G(u, v)$ between any pair of vertices $u, v \in V$ can be approximated by their respective distance $d_{\mathcal{T}}(u, v)$ in $\mathcal{T}$ within an (expected) factor of $\mathcal{O}(\log n)$.

Remark 2. *We note that traditionally HSTs are neither balanced nor are required to have weights that are specifically powers of 2. However, we can transform any general HST into our specific definition, and this has been accounted for in the statement of the above theorem. The details are deferred to Section B of the Appendix.*

3 Overview of our approach

In this section we present the key steps of our approach towards designing the $\mathcal{O}(\log n)$-regret online learning algorithm for Problem 1. Our approach can be summarized in the following three pillars:

1. In Section 3.1 we introduce a *fractional relaxation* of Problem 1 in the special case of HSTs (Problem 2). Problem 2 is an artificial problem at which the learner can place a *fractional amount of facility* to the leaves of an HST so as to fractionally serve the arrived clients. Since the *optimal static solution* of Problem 2 lower bounds the *optimal static solution* of Problem 1 in the special case of HSTs, the first step of our approach is to design an $\mathcal{O}(1)$-regret algorithm for Problem 2.
2. In Section 3.2 we present the formal guarantees of a novel randomized rounding scheme, called $\mathrm{Cut\&Round}$, that is client-oblivious and converts any *fractional solution* for Problem 2 into an actual placement of k facilities on the leaves of the HST with just an $\mathcal{O}(1)$-overhead in the connection and the moving cost.
3. In Section 3.3 we present how the *fractional algorithm* for Problem 2 together with the $\mathrm{Cut\&Round}$ rounding naturally lead to an $\mathcal{O}(1)$-regret online learning algorithm for Problem 1 in the special case of HSTs (Algorithm 1). Our main algorithm, presented in Algorithm 2, then consists of running Algorithm 1 into an $\mathcal{O}(\log n)$ HST embedding of input graph.

3.1 A Fractional Relaxation for HSTs

In this section we introduce a fractional relaxation for Problem 1, called *Fractional k-Clustering with Moving Costs on HSTs* (Problem 2). Fix any HST $\mathcal{T}(V, E, w)$ (in this section, V denotes the nodes of the HST). We begin by presenting a *fractional extension* of placing k facilities on the leaves of $\mathcal{T}$.

Definition 6. *The set of fractional facility placements $\mathcal{FP}(\mathcal{T})$ consists of all vectors $y \in \mathbb{R}^{|V|}$ such that*

1. $y_v \in [0, 1]$ *for all leaves $v \in L(\mathcal{T})$.*
2. $y_v = \sum_{u \in \mathrm{cld}(v)} y_u$ *for all non-leaves $v \notin L(\mathcal{T})$.*
3. $\sum_{v \in L(\mathcal{T})} y_v = k$, *i.e. the total amount of facility on the leaves equals k.*

For a leaf vertex $v \in L(\mathcal{T})$, y_v simply denotes the fractional amount of facilities that are placed on it. For all non-leaf vertices $v \notin L(\mathcal{T})$, y_v denotes the total amount of facility placed in the leaves of the sub-tree $T(v)$. Thus, any integral vector $y \in \mathcal{FP}(\mathcal{T}) \cap \mathbb{N}$ corresponds to a placement of k facilities on the leaves of $\mathcal{T}$.

In Definitions 7 and 8 we extend the notion of connection and moving cost for fractional facility placements. In the special case of integral facility placements, Definitions 7 and 8 respectively collapse to Definitions 1 and 2 (a formal proof is given in Claims 1 and 2 of Section C of the Appendix).

Definition 7. *The fractional connection cost of a set of clients $R \subseteq L(\mathcal{T})$ under $y \in \mathcal{FP}(\mathcal{T})$ is defined as*

$$f_R(y) := \sum_{j \in R} \sum_{v \in P(j,r)} 2^{lev(v)+1} \cdot \max(0, 1 - y_v)$$

where $P(j, r)$ denotes the set of vertices in the (unique) path from the leaf $j \in L(\mathcal{T})$ to the root r.

Definition 8. *The fractional moving cost between any $y, y' \in \mathcal{FP}(\mathcal{T})$ is defined as*

$$||y - y'||_{\mathcal{T}} := \gamma \cdot \sum_{v \in V(\mathcal{T})} 2^{lev(v)} \cdot |y_v - y'_v|$$

We are now ready to present our fractional generalization of Problem 1 in the special case of HSTs.

Problem 2 (*Fractional k-Clustering with Moving Costs on HSTs*)**.** *Fix any HST $\mathcal{T}$. At each round $t = 1, \ldots, T$:*

1. *The learner selects a vector $y^t \in \mathcal{FP}(\mathcal{T})$.*
2. *The adversary selects a set of clients $R_t \subseteq L(\mathcal{T})$.*
3. *The learner suffers cost $f_{R_t}(y^t) + ||y^t - y^{t-1}||_{\mathcal{T}}$.*

In Section 4, we develop and present an $\mathcal{O}(1)$-regret algorithm for Problem 2 (see Algorithm 3). To this end, we present its formal regret guarantee established in Theorem 3.

Theorem 3. *There exists a polynomial-time online learning algorithm for Problem 2 (Algorithm 3), such that for any sequence $R_1, \ldots, R_T \subseteq L(\mathcal{T})$, its output $y^1, \ldots, y^T$ satisfies*

$$\sum_{t=1}^{T} f_{R_t}(y^t) + \sum_{t=2}^{T} ||y^t - y^{t-1}||_{\mathcal{T}} \leq \frac{3}{2} \cdot \min_{y^* \in \mathcal{FP}(\mathcal{T})} \sum_{t=1}^{T} f_{R_t}(y^*) + \beta \cdot \sqrt{T}$$

for $\beta = \mathcal{O}\left(k \cdot |L(\mathcal{T})|^{3/2} \cdot D_{\mathcal{T}} \cdot \max(\gamma, 1)\right)$.

3.2 From Fractional to Integral Placements in HSTs

As already mentioned, the basic idea of our approach is to convert at each round $t \geq 1$ the *fractional placement* $y^t \in \mathcal{FP}(\mathcal{T})$ produced by Algorithm 3 into an integral facility placement $F_t \subseteq L(\mathcal{T})$ with $|F_t| = k$ on the leaves of the HST. In order to guarantee small regret, our rounding scheme should preserve both the connection and the moving cost of the fractional solution within constant factors for *any possible set of arriving clients*. In order to guarantee the latter, our rounding scheme $\mathrm{Cut\&Round}$ (Algorithm 4) uses shared randomness across different rounds. $\mathrm{Cut\&Round}$ is rather complicated and is presented in Section 5. To this end, we present its formal guarantee.

Theorem 4. *There exists a linear-time deterministic algorithm, called* $\mathrm{Cut\&Round}$ *(Algorithm 4), that takes as input an HST $\mathcal{T}$, a fractional facility placement $y \in \mathcal{FP}(\mathcal{T})$ and a vector $\alpha \in [0,1]^{|V|}$ and outputs a placement of k facilities $F \leftarrow \mathrm{Cut\&Round}(\mathcal{T}, y, \alpha)$ on the leaves of $\mathcal{T}$ ($F \subseteq L(\mathcal{T})$ and $|F| = k$) such that*

1. *$\mathrm{E}_{\alpha \sim \mathrm{Unif}(0,1)}[C_R(F)] = f_R(y)$ for all client requests $R \subseteq L(\mathcal{T})$.*
2. *$\mathrm{E}_{\alpha \sim \mathrm{Unif}(0,1)}[\gamma \cdot M_{\mathcal{T}}(F, F')] \leq 4 \cdot ||y - y'||_{\mathcal{T}}$ for all other fractional facility placements $y' \in \mathcal{FP}(\mathcal{T})$ and $F' \leftarrow \mathrm{Cut\&Round}(\mathcal{T}, y', \alpha)$.*

Item 1 of Theorem 4 establishes that although $\mathrm{Cut\&Round}$ is *oblivious* to the arrived set of clients $R_t \subseteq L(\mathcal{T})$, the expected connection cost of the output equals the *fractional connection cost* under $y^t \in \mathcal{FP}(\mathcal{T})$. Item 2 of Theorem 4 states that once the same random seed α is used into two consecutive time steps, then the expected moving cost between the facility-placements F_t and F_{t+1} is at most $\mathcal{O}(1)$-times the fractional moving cost between y^t and y^{t+1}. Both properties crucially rely on the structure of the HST and consist one of the main technical contributions of our work.

3.3 Overall Online Learning Algorithm

We are now ready to formally introduce our main algorithm (Algorithm 2) and prove Theorem 1. First, we combine the algorithms from Theorems 3 and 4 to design an $\mathcal{O}(1)$-regret algorithm for Problem 1 on HSTs (Algorithm 1). Up next we present how Algorithm 1 can be converted into an $\mathcal{O}(\log n)$-regret online learning algorithm for general graphs, using the metric embedding technique of Theorem 2, resulting to our final algorithm (Algorithm 2).

Algorithm 1 $\mathcal{O}(1)$-regret for HSTs.

1: **Input:** A sequence $R_1, \ldots, R_T \subseteq L(\mathcal{T})$.
2: The learner samples $\alpha_v \sim \mathrm{Unif}(0,1)$ for all $v \in V(\mathcal{T})$.
3: **for** each round $t = 1$ **to** T **do**
4: The learner places the k facilities to the leaves of the HST $\mathcal{T}$ based on the output $F_t := \mathrm{Cut\&Round}(\mathcal{T}, y^t, \alpha)$.
5: The learner learns $R_t \subseteq L(\mathcal{T})$.
6: The learner updates $y^{t+1} \in \mathcal{FP}(\mathcal{T})$ by running Algorithm 3 for Problem 2 with input $R_1, \ldots, R_t$.
7: **end for**

Algorithm 2 $\mathcal{O}(\log n)$-regret for graphs.

1: **Input:** A sequence $R_1, \ldots, R_T \subseteq L(\mathcal{T})$.
2: The learner embeds $G(V, E, w)$ into a (random) HST $\mathcal{T}$ with $L(\mathcal{T}) = V$ via the procedure of Theorem 2.
3: **for** each round $t = 1$ **to** T **do**
4: The learner selects a facility-placement $F_t \subseteq V$.
5: The learner learns $R_t \subseteq V$.
6: The learner updates F_{t+1} by giving as input $R_1, \ldots, R_t \subseteq L(\mathcal{T})$ to Algorithm 1 for $\mathcal{T}$.
7: **end for**

Theorem 5. *For any sequence of client requests $R_1, \ldots, R_T \subseteq L(\mathcal{T})$, the sequence of facility-placements $F_1, \ldots, F_T \subseteq L(\mathcal{T})$ produced by Algorithm 1 satisfies*

$$\mathbb{E}\left[\sum_{t=1}^{T} C_{R_t}(F_t) + \gamma \cdot \sum_{t=2}^{T} M_{\mathcal{T}}(F_t, F_{t-1})\right] \le 6 \cdot \min_{|F^*|=k} \sum_{t=1}^{T} C_{R_t}(F^*) + \beta \cdot \sqrt{T}$$

for $\beta = \mathcal{O}\left(k \cdot |L(\mathcal{T})|^{3/2} \cdot D_{\mathcal{T}} \cdot \max(\gamma, 1)\right)$.

Theorem 5 establishes that Algorithm 1 achieves constant regret in the special case of HSTs and its proof easily follows by Theorems 3 and 4. Then, the proof of Theorem 1 easily follows by Theorem 2 and Theorem 5. All the proofs are deferred to Section C of the Appendix.

4 $\mathcal{O}(1)$-Regret for Fractional HST Clustering

In this section we present the $\mathcal{O}(1)$-regret algorithm for Problem 2, described in Algorithm 3, and exhibit the key ideas in establishing Theorem 3. Without loss of generality, we can assume that the facility-weight satisfies $\gamma \ge 1$[3].

Algorithm 3 is the well-known online learning algorithm *Follow the Regularized Leader* (FTRL) with a specific regularizer $R_{\mathcal{T}}(\cdot)$ presented in Definition 9. Our results crucially rely on the properties of this regularizer since it takes into account the HST structure and permits us to bound the fractional moving cost of FTRL.

Definition 9. *Given an HST $\mathcal{T}$, the dilated entropic regularizer $R_{\mathcal{T}}(y)$ over $y \in \mathcal{FP}(\mathcal{T})$ is defined as*

$$R_{\mathcal{T}}(y) := \sum_{v \neq r} 2^{\mathrm{lev}(v)} \cdot (y_v + \delta_v) \cdot \ln\left(\frac{y_v + \delta_v}{y_{p(v)} + \delta_{p(v)}}\right)$$

where $\delta_v := (k/n) \cdot |L(\mathcal{T}) \cap T(v)|$ and $n := |L(\mathcal{T})|$.

Algorithm 3 selects at each step t the facility placement $y^t \in \mathcal{FP}(\mathcal{T})$ that minimizes a convex combination of the total fractional connection cost for the sub-sequence $R_1, \ldots, R_{t-1}$ and $R_{\mathcal{T}}(y)$. The regularization term ensures the stability of the output, which will result in a bounded fractional moving cost.

[3] If not, establishing our guarantees for $\gamma = 1$ will clearly upper bound the actual moving cost.

Algorithm 3 FTRL with dilated entropic regularization

1. **Input:** An adversarial sequence $R_1, \dots, R_T \subseteq L(\mathcal{T})$.
2. **for** $t = 1$ **to** T **do**
3. The learner selects $y^t \in \mathcal{FP}(\mathcal{T})$.
4. The learner suffers cost $f_{R_t}(y^t) + ||y^t - y^{t-1}||_\mathcal{T}$.
5. The learner updates $y^{t+1} \leftarrow \arg\min_{y \in \mathcal{FP}(\mathcal{T})} \left[\sum_{s=1}^t f_{R_s}(y) + (\gamma\sqrt{nT}) \cdot R_\mathcal{T}(y)\right]$.
6. **end for**

Analysis of Algorithm 3. Due to space limitations, all proofs are moved to Section D of the Appendix. The primary reason for the specific selection of the regularizer at Definition 9 is that $R_\mathcal{T}(\cdot)$ is strongly convex with respect to the norm $|| \cdot ||_\mathcal{T}$ of Definition 8, as established in Lemma 1 which is the main technical contribution of the section. We use $D = D_\mathcal{T}$ for the diameter of $\mathcal{T}$.

Lemma 1. *For any vectors $y, y' \in \mathcal{FP}(\mathcal{T})$,*

$$R_\mathcal{T}(y') \geq R_\mathcal{T}(y) + \langle \nabla R_\mathcal{T}(y), y' - y\rangle + \left(8kD\gamma^2\right)^{-1} \cdot ||y - y'||^2_\mathcal{T}$$

The strong convexity of $R_\mathcal{T}(y)$ with respect to $|| \cdot ||_\mathcal{T}$ is crucial since it permits us to bound the moving cost of Algorithm 3 by its fractional connection cost.

Lemma 2. *For any sequence $R_1, \dots, R_T \subseteq L(\mathcal{T})$, the output of Algorithm 3 satisfies*

$$\sum_{t=2}^T ||y^t - y^{t-1}||_\mathcal{T} \leq \frac{1}{2} \cdot \sum_{t=1}^T f_{R_t}(y^t) + \mathcal{O}\left(\gamma k D\right) \cdot \sqrt{T}$$

We remark that using another regularizer $R(\cdot)$ that is strongly convex with respect to another norm $|| \cdot ||$ would still imply Lemma 1 with respect to $|| \cdot ||$. The problem though is that the *fractional moving cost* $\sum_{t=1}^T ||y_t - y_{t-1}||$ can no longer be associated with the actual moving cost $\sum_{t=1}^T M_\mathcal{T}(F_t, F_{t-1})$. It is for this reason that using a regularizer that is strongly convex with respect to $|| \cdot ||_\mathcal{T}$ is crucial.

Next, by adapting the standard analysis of FTRL to our specific setting, we derive Lemma 3 establishing that Algorithm 3 admits bounded connection cost.

Lemma 3. *For any sequence $R_1, \dots, R_T \subseteq L(\mathcal{T})$, the output of Algorithm 3 satisfies*

$$\sum_{t=1}^T f_{R_t}(y^t) \leq \min_{y^* \in \mathcal{FP}} \sum_{t=1}^T f_{R_t}(y^*) + \mathcal{O}\left(kn^{3/2}D\gamma\right) \cdot \sqrt{T}$$

The proof of Theorem 3 directly follows by Lemma 2 and 3. We conclude the section by presenting how Step 5 of Algorithm 3 can be efficiently implemented, namely

$$\min_{y \in \mathcal{FP}(\mathcal{T})} \Phi_t(y) := \sum_{s=1}^t f_{R_s}(y) + (\gamma\sqrt{nT}) \cdot R_\mathcal{T}(y).$$

Since $\Phi_t(y)$ is strongly convex and the set $\mathcal{FP}(\mathcal{T})$ is a polytope, one could use standard optimization algorithms such as the *ellipsoid method* or *projected gradient descent* to approximately minimize $\Phi_t(y)$ given access to a *sub-gradient oracle for* $\Phi_t(\cdot)$. In Claim 11 of Section D of the Appendix, we establish that the sub-gradients of $\Phi(\cdot)$ can be computed in polynomial time and thus any of the previous methods can be used to approximately minimize $\Phi(\cdot)$. In Lemma 4 we establish the intuitive fact that approximately implementing Step 5 does not affect the guarantees of Theorem 3.

Lemma 4. *Let y^t be the minimizer of $\Phi_t(\cdot)$ in $\mathcal{FP}(T)$ and let $z^t \in \mathcal{FP}(\mathcal{T})$ be any point such that $\Phi_t(z^t) \leq \Phi_t(y^t) + \epsilon$ for some $\epsilon = \mathcal{O}(T^{-1/2})$. Then,*

$$f_{R_t}(z^t) + ||z^t - z^{t-1}||_\mathcal{T} \leq f_{R_t}(y^t) + ||y^t - y^{t-1}||_\mathcal{T} + \mathcal{O}\left(kn^{3/2}D\gamma\right) \cdot T^{-1/2}$$

Remark 3. *In our implementation of the algorithm, we approximately solve Step 5 of Algorithm 3 via Mirror Descent based on the Bregman divergence of $\mathcal{R}_\mathcal{T}(\cdot)$. This admits the same convergence rates as projected gradient descent but the projection step can be computed in linear time with respect to the size of the HST $\mathcal{T}$. We present the details of our implementation in Section C of the Appendix.*

5 The Cut&Round Rounding

In this section we present our novel rounding scheme (Algorithm Cut&Round) as well as the main steps that are required in order to establish Theorem 4. To ease notation, for any real number $x \geq 0$ we denote its decimal part as $\delta(x) = x - \lfloor x \rfloor$.

Algorithm 4 Cut&Round.

1: **Input:** An HST $\mathcal{T}$, a fractional placement $y \in \mathcal{FP}(\mathcal{T})$ and thresholds $\alpha_v \in [0, 1]$ for all $v \in V(\mathcal{T})$.
2: $Y_r \leftarrow k$
3: **for** levels $\ell = h(\mathcal{T})$ **to** 1 **do**
4:     **for** all nodes v with $lev(v) = \ell$ **do**
5:         $Y_{rem} \leftarrow Y_v$
6:         $y_{rem} \leftarrow y_v$
7:         **for** all children $u \in \text{cld}(v)$ **do**
8:             $Y_u \leftarrow \text{Alloc}(y_u, Y_{rem}, y_{rem}, \alpha_u)$
9:             $Y_{rem} \leftarrow Y_{rem} - Y_u$
10:             $y_{rem} \leftarrow y_{rem} - y_u$
11:         **end for**
12:     **end for**
13: **end for**
14: **return** $F := \{u \in L(\mathcal{T}) : Y_u = 1\}$.

Algorithm 5 Alloc.

Input: Numbers $y_u, y_{rem} \geq 0, Y_{rem} \in \mathbb{N}$ and $\alpha_u \in [0, 1]$.
if $Y_{rem} == \lfloor y_{rem} \rfloor$ **then**
    if $\delta(y_u) < \delta(y_{rem})$ **then**
        $Y_u \leftarrow \lfloor y_u \rfloor$
    else
        $Y_u \leftarrow \lfloor y_u \rfloor + \mathbb{1}\left[a_u \leq \frac{\delta(y_u) - \delta(y_{rem})}{1 - \delta(y_{rem})}\right]$
    end if
else
    if $\delta(y_u) < \delta(y_{rem})$ **then**
        $Y_u \leftarrow \lfloor y_u \rfloor + \mathbb{1}\left[a_u \leq \frac{\delta(y_u)}{\delta(y_{rem})}\right]$
    else
        $Y_u \leftarrow \lfloor y_u \rfloor + 1$
    end if
end if
Return Y_u.

On principle, Cut&Round (Algorithm 4) assigns to each vertex v an integer number of facilities Y_v to be placed at the leaves of its sub-tree. Notice that due to sub-routine Alloc (Algorithm 5), Y_v either equals $\lfloor y_v \rfloor$ or $\lfloor y_v \rfloor + 1$. Cut&Round initially assigns k facilities to the set of leaves that descend from the root r, which is precisely $L(\mathcal{T})$. Then, it moves in decreasing level order to decide Y_v for each node v. Once Y_v is determined (Step 5), the Y_v facilities are allocated to the sub-trees of its children $u \in \text{cld}(v)$ (Steps 7-10) via sub-routine Alloc using the thresholds α_u, in a manner that guarantees that $Y_v = \sum_{u \in \text{cld}(v)} Y_u$ (see Section E.1 of the Appendix). This implies the feasibility of Cut&Round, as exactly k facilities are placed in the leaves of $\mathcal{T}$ at the end of the process.

Assuming that the set of thresholds α_v is randomly drawn from the uniform distribution in $[0, 1]$, sub-routine Alloc (Algorithm 5) guarantees that Y_v either equals $\lfloor y_v \rfloor$ or $\lfloor y_v \rfloor + 1$ while $\mathbb{E}_\alpha [Y_v] = y_v$. This is formally captured in Lemma 5 and is crucial in the proof of Theorem 4.

Lemma 5. *Consider Algorithm 4 given as input a vector $y \in \mathcal{FP}(\mathcal{T})$ and random thresholds $\alpha_v \sim \text{Unif}(0, 1)$. Then,*

$$Y_v = \begin{cases} \lfloor y_v \rfloor & \text{with probability } 1 - \delta(y_v) \\ \lfloor y_v \rfloor + 1 & \text{with probability } \delta(y_v) \end{cases}$$

By coupling Lemma 5 with the HST structure we are able to establish Theorem 4. The proof is technically involved and thus deferred to Section E of the Appendix.

6 Conclusion

In this work, we designed the first polynomial-time online learning algorithm for *Online k-Clustering with Moving Costs* that achieves $\mathcal{O}(\log n)$-regret with respect to the cost of the optimal *static* facility placement, extending the results of Fotakis et al. [30] for the special case of $\gamma = 0$. A interesting future direction is to investigate whether a polynomial-time online learning algorithm with $\mathcal{O}(1)$-regret for the problem is theoretically possible or not.

Limitations: Our current optimality guarantees are with respect to the optimal *static* facility placement. Going beyond the notion of regret, an intriguing future direction is establishing guarantees with respect to the *optimal dynamic facility-placement* that moves facilities from round to round by suffering the corresponding moving cost.

References

[1] Alekh Agarwal, Daniel J. Hsu, Satyen Kale, John Langford, Lihong Li, and Robert E. Schapire. Taming the monster: A fast and simple algorithm for contextual bandits. In *International Conference on Machine Learning*, 2014.

[2] Nir Ailon. Improved bounds for online learning over the permutahedron and other ranking polytopes. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics*, AISTATS 2014, 2014.

[3] Soroush Alamdari and David B. Shmoys. A bicriteria approximation algorithm for the k-center and k-median problems. In *Workshop on Approximation and Online Algorithms*, 2017.

[4] Hyung-Chan An, Ashkan Norouzi-Fard, and Ola Svensson. Dynamic facility location via exponential clocks. *ACM Trans. Algorithms*, 13(2):21:1–21:20, 2017.

[5] Tarique Anwar, Surya Nepal, Cecile Paris, Jian Yang, Jia Wu, and Quan Z. Sheng. Tracking the evolution of clusters in social media streams. *IEEE Transactions on Big Data*, pages 1–15, 2022.

[6] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristic for k-median and facility location problems. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, STOC '01, page 21–29. Association for Computing Machinery, 2001.

[7] Baruch Awerbuch and Robert Kleinberg. Online linear optimization and adaptive routing. *J. Comput. Syst. Sci.*, 2008.

[8] Maria-Florina Balcan and Avrim Blum. Approximation algorithms and online mechanisms for item pricing. In *ACM Conference on Electronic Commerce*, 2006.

[9] Nikhil Bansal, Niv Buchbinder, Aleksander Madry, and Joseph Naor. A polylogarithmic-competitive algorithm for the k-server problem. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 267–276. IEEE Computer Society, 2011.

[10] Yair Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. *Proceedings of 37th Conference on Foundations of Computer Science*, pages 184–193, 1996.

[11] Avrim Blum and Carl Burch. On-line learning and the metrical task system problem. *Mach. Learn.*, 39(1):35–58, 2000.

[12] Sébastien Bubeck, Michael B. Cohen, Yin Tat Lee, James R. Lee, and Aleksander Madry. k-server via multiscale entropic regularization. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 3–16. ACM, 2018.

[13] Moses Charikar and Sudipto Guha. Improved combinatorial algorithms for the facility location and k-median problems. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, FOCS '99. IEEE Computer Society, 1999.

[14] Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys. A constant-factor approximation algorithm for the k-median problem (extended abstract). In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, STOC '99, page 1–10. Association for Computing Machinery, 1999.

[15] Moses Charikar and Shi Li. A dependent lp-rounding approach for the k-median problem. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012*, volume 7391 of *Lecture Notes in Computer Science*, pages 194–205. Springer, 2012.

[16] Vincent Cohen-Addad, Benjamin Guedj, Varun Kanade, and Guy Rom. Online k-means clustering. In Arindam Banerjee and Kenji Fukumizu, editors, *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, volume 130 of *Proceedings of Machine Learning Research*, pages 1126–1134. PMLR, 2021.

[17] Aaron Cote, Adam Meyerson, and Laura J. Poplawski. Randomized k-server on hierarchical binary trees. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 227–234. ACM, 2008.

[18] Bart de Keijzer and Dominik Wojtczak. Facility reallocation on the line. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018*, pages 188–194, 2018.

[19] Sina Dehghani, Soheil Ehsani, MohammadTaghi Hajiaghayi, Vahid Liaghat, and Saeed Seddighin. Stochastic k-server: How should uber work? In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 126:1–126:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[20] Sina Dehghani, MohammadTaghi Hajiaghayi, Hamid Mahini, and Saeed Seddighin. Price of competition and dueling games. *arXiv preprint arXiv:1605.04004*, 2016.

[21] Miroslav Dudík, Nika Haghtalab, Haipeng Luo, Robert E. Schapire, Vasilis Syrgkanis, and Jennifer Wortman Vaughan. Oracle-efficient online learning and auction design. In *58th IEEE Annual Symposium on Foundations of Computer Science*, FOCS 2017, 2017.

[22] Miroslav Dudík, Daniel J. Hsu, Satyen Kale, Nikos Karampatziakis, John Langford, Lev Reyzin, and Tong Zhang. Efficient optimal learning for contextual bandits. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, UAI 2011, 2011.

[23] David Eisenstat, Claire Mathieu, and Nicolas Schabanel. Facility location in evolving metrics. In *Automata, Languages, and Programming - 41st International ColloquiumICALP 2014*, volume 8573 of *Lecture Notes in Computer Science*, pages 459–470. Springer, 2014.

[24] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '03, page 448–455, New York, NY, USA, 2003. Association for Computing Machinery.

[25] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Syst. Sci.*, 69(3):485–497, 2004.

[26] Gabriele Farina, Christian Kroer, and Tuomas Sandholm. Optimistic regret minimization for extensive-form games via dilated distance-generating functions. In *Neural Information Processing Systems*, 2019.

[27] Hendrik Fichtenberger, Silvio Lattanzi, Ashkan Norouzi-Fard, and Ola Svensson. Consistent k-clustering for general metrics. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 2660–2678. SIAM, 2021.

[28] Dimitris Fotakis, Loukas Kavouras, Panagiotis Kostopanagiotis, Philip Lazos, Stratis Skoulakis, and Nikos Zarifis. Reallocating multiple facilities on the line. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019*, pages 273–279, 2019.

[29] Dimitris Fotakis, Thanasis Lianeas, Georgios Piliouras, and Stratis Skoulakis. Efficient online learning of optimal rankings: Dimensionality reduction via gradient descent. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020.

[30] Dimitris Fotakis, Georgios Piliouras, and Stratis Skoulakis. Efficient online learning for dynamic k-clustering. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 3396–3406. PMLR, 18–24 Jul 2021.

[31] Takahiro Fujita, Kohei Hatano, and Eiji Takimoto. Combinatorial online prediction via metarounding. In *24th International Conference on Algorithmic Learning Theory*, ALT 2013, 2013.

[32] Dan Garber. Efficient online linear optimization with approximation algorithms. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS 2017, 2017.

[33] Xiangyu Guo, Janardhan Kulkarni, Shi Li, and Jiayi Xian. Consistent k-median: Simpler, better and robust. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 1135–1143. PMLR, 13–15 Apr 2021.

[34] Elad Hazan. Introduction to online convex optimization. *CoRR*, abs/1909.05207, 2019.

[35] Elad Hazan, Wei Hu, Yuanzhi Li, and Zhiyuan Li. Online improper learning with an approximation oracle. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[36] Elad Hazan and Satyen Kale. Online submodular minimization. *J. Mach. Learn. Res.*, 2012.

[37] Elad Hazan, Satyen Kale, and Shai Shalev-Shwartz. Near-optimal algorithms for online matrix prediction. In *25th Annual Conference on Learning Theory*, COLT 2012, 2012.

[38] Elad Hazan and Tomer Koren. The computational power of optimization in online learning. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing*, STOC 2016, 2016.

[39] David P. Helmbold, Robert E. Schapire, and M. Long. Predicting nearly as well as the best pruning of a decision tree. In *Machine Learning*, 1997.

[40] David P. Helmbold and Manfred K. Warmuth. Learning permutations with exponential weights. In *Proceedings of the 20th Annual Conference on Learning Theory*, COLT 2007, 2007.

[41] Nicole Immorlica, Adam Tauman Kalai, Brendan Lucier, Ankur Moitra, Andrew Postlewaite, and Moshe Tennenholtz. Dueling algorithms. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, STOC '11, page 215–224, New York, NY, USA, 2011. Association for Computing Machinery.

[42] Kamal Jain, Mohammad Mahdian, and Amin Saberi. A new greedy approach for facility location problems. In John H. Reif, editor, *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 731–740. ACM, 2002.

[43] Kamal Jain and Vijay V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *J. ACM*, 48(2):274–296, 2001.

[44] Stefanie Jegelka and Jeff A. Bilmes. Online submodular minimization for combinatorial structures. In *Proceedings of the 28th International Conference on Machine Learning*, ICML 2011, 2011.

[45] Sham Kakade, Adam Tauman Kalai, and Katrina Ligett. Playing games with approximation algorithms. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing*, STOC 2007, 2007.

[46] Adam Kalai and Santosh Vempala. Efficient algorithms for online decision problems. In *J. Comput. Syst. Sci*. Springer, 2003.

[47] Wouter M. Koolen, Manfred K. Warmuth, and Jyrki Kivinen. Hedging structured concepts. In *the 23rd Conference on Learning Theory, COLT 2010*, 2010.

[48] Elias Koutsoupias. The k-server problem. *Comput. Sci. Rev.*, 3(2):105–118, 2009.

[49] Elias Koutsoupias and Christos H. Papadimitriou. On the k-server conjecture. *J. ACM*, 42(5):971–983, 1995.

[50] Amit Kumar. Constant factor approximation algorithm for the knapsack median problem. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, page 824–832. Society for Industrial and Applied Mathematics, 2012.

[51] Amit Kumar, Yogish Sabharwal, and Sandeep Sen. Linear-time approximation schemes for clustering problems in any dimensions. *J. ACM*, 57(2), 2010.

[52] Silvio Lattanzi and Sergei Vassilvitskii. Consistent k-clustering. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1975–1984. PMLR, 2017.

[53] Shi Li and Ola Svensson. Approximating k-median via pseudo-approximation. *SIAM J. Comput.*, 45(2):530–547, 2016.

[54] Jyh-Han Lin and Jeffrey Scott Vitter. Approximation algorithms for geometric median problems. *Information Processing Letters*, 44(5):245 – 249, 1992.

[55] M.E.J. Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003.

[56] Romualdo Pastor-Satorras and Alessandro Vespignani. Epidemic Spreading in Scale-Free Networks. *Physical Review Letters*, 86(14):3200–3203, 2001.

[57] Holakou Rahmanian and Manfred K. Warmuth. Online dynamic programming. In *NIPS*, 2017.

[58] Juliette Stehlé, Nicolas Voirin, Alain Barrat, Ciro Cattuto, Lorenzo Isella, Jean-François Pinton, Marco Quaggiotto, Wouter Van den Broeck, Corinne Régis, Bruno Lina, and Philippe Vanhems. High-resolution measurements of face-to-face contact patterns in a primary school. *PLOS ONE*, 6(8), 2011.

[59] Matthew J. Streeter and Daniel Golovin. An online algorithm for maximizing submodular functions. In *22nd Annual Conference on Neural Information Processing Systems*, NIPS 2008, 2008.

[60] Daiki Suehiro, Kohei Hatano, Shuji Kijima, Eiji Takimoto, and Kiyohito Nagano. Online prediction under submodular constraints. In *Algorithmic Learning Theory*, ALT 2012, 2012.

[61] Eiji Takimoto and Manfred K. Warmuth. Predicting nearly as well as the best pruning of a planar decision graph. In *Theoretical Computer Science*, 2000.

[62] Eiji Takimoto and Manfred K. Warmuth. Path kernels and multiplicative updates. *J. Mach. Learn. Res.*, 2003.

[63] Chayant Tantipathananandh, Tanya Berger-Wolf, and David Kempe. A framework for community identification in dynamic social networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, page 717–726. Association for Computing Machinery, 2007.

[64] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, USA, 1st edition, 2011.

[65] Shota Yasutake, Kohei Hatano, Shuji Kijima, Eiji Takimoto, and Masayuki Takeda. Online linear optimization over permutations. In *Proceedings of the 22nd International Conference on Algorithms and Computation*, ISAAC 2011, 2011.

[66] Neal E. Young. K-medians, facility location, and the chernoff-wald bound. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '00, page 86–95. Society for Industrial and Applied Mathematics, 2000.