Smoothed Analysis of Online Metric Problems

Christian Coester
Department of Computer Science, University of Oxford, UK

Jack Umenberger
Department of Engineering Science, University of Oxford, UK

Abstract

We study three classical online problems – k-server, k-taxi, and chasing size k sets – through a lens of smoothed analysis. Our setting allows request locations to be adversarial up to small perturbations, interpolating between worst-case and average-case models. Specifically, we show that if the metric space is contained in a ball in any normed space and requests are drawn from distributions whose density functions are upper bounded by $1/\sigma$ times the uniform density over the ball, then all three problems admit polylog(k/σ)-competitive algorithms. Our approach is simple: it reduces smoothed instances to fully adversarial instances on finite metrics and leverages existing algorithms in a black-box manner. We also provide a lower bound showing that no algorithm can achieve a competitive ratio sub-polylogarithmic in k/σ, matching our upper bounds up to the exponent of the polylogarithm. In contrast, the best known competitive ratios for these problems in the fully adversarial setting are $2k-1$, ∞ and $\Theta(k^2)$, respectively.

2012 ACM Subject Classification Theory of computation → Online algorithms; Theory of computation → Design and analysis of algorithms

Keywords and phrases Online Algorithms, Competitive Analysis, Smoothed Analysis, k-server, k-taxi, Metrical Service Systems

Digital Object Identifier 10.4230/LIPIcs.ESA.2025.115

Funding *Christian Coester*: Funded by the European Union (ERC grant CCOO 101165139). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.
Jack Umenberger: Supported by the University of Oxford's Strategic Research Fund through the ZERO Institute.

1 Introduction

In online algorithms, an algorithm has to make decisions online while the input is being revealed, without knowledge of the future input. Traditionally, the majority of research in the field has analyzed problems in the worst-case regime, in terms of the *competitive ratio*, i.e., the largest possible ratio between the algorithm's performance and the optimum in hindsight. This perspective enables clean mathematical models and robust guarantees for algorithms that are satisfied regardless of the input they face. However, worst-case instances are often degenerate and unlikely to occur in practice. Tuning algorithms against such pathological scenarios can lead to worst-case-like performance even when actual inputs are much more favorable.

Moreover, the guarantees that can be achieved even under the most pessimistic assumptions are often weak by necessity. While tight competitive ratio bounds can appeal theoretically, algorithms are unlikely to be adopted in practice if the best promise is that they perform at most some large (e.g., polynomial or exponential) factor worse than optimal. Worse yet, for some problems such as the k-taxi problem, no finite bound on the competitive ratio is currently known for general metric spaces of infinitely many points, even if their diameter is bounded.

The research community generally understands the limitations of worst-case analysis, and there have been many demands for and an increased interest in beyond-worst-case analysis in recent years [46]. A major success story of beyond-worst-case analysis has been Spielman and Teng's smoothed analysis [48], spectacularly explaining why the simplex algorithm for linear problems performs well in practice despite its exponential worst-case running time. They show that perturbing any instance by small Gaussian noise yields an instance which the simplex algorithm solves in expected polynomial time. By varying the size of the random perturbations, smoothed analysis allows to interpolate between worst-case and average-case settings. Other examples of smoothed analysis of the running time of offline problems include k-means [3, 2], TSP [23, 41, 24], local max-cut [1] and more general local search problems [29]. We refer to [46] for more background.

In the context of online problems, prior smoothed analysis has focused mostly on regret minimization in online learning [45, 33, 20, 37, 44, 35, 36, 7, 34, 22] and online discrepancy minimization [36, 4, 5]. In contrast, understanding is much more limited about the smoothed complexity of more dynamic online problems which are typically analyzed through their competitive ratio (rather than regret). The only examples we are aware of are the work of [6], which conduct smoothed competitive analysis of the multi-level feedback algorithm for non-clairvoyant scheduling, and [47], who analyze the work function algorithm for metrical task systems when cost function values are randomly perturbed.

Smoothed Competitive Analysis in Metric Spaces. Here, we initiate smoothed competitive analysis of online problems in metric spaces with randomly perturbed request locations. To do so, we study three classical online problems: namely, the *k-server* problem [40], its generalization the *k-taxi* problem [27, 18], and *chasing small sets* [17] (also known as metrical service systems, and equivalent to layered graph traversal [43, 26]). In these problems, the goal is to minimize movement in a metric space while serving a sequence of requests. In the k-server problem, the algorithm controls k servers and each request is a location that one of the servers must move to. The k-taxi problem generalizes this to the case where each request is a pair of locations, modelling the pick-up and drop-off location of a ride request. In chasing small sets, there is a single server, and each request is a set of up to k points that the server must move to.

On general metric spaces, the best known competitive ratio is $\Theta(k)$ for the k-server problem [38] and $\Theta(k^2)$ for chasing small sets [11]. For the k-taxi problem, the situation is even worse and it is not even known whether any finite competitive ratio can be achieved, even for $k = 4$ servers on a line metric. Better bounds exist only in special cases such as *finite* metrics of n points or bounded aspect ratio Δ, where the k-server problem admits competitive ratios of $O(\log n \log^2 k)$ and $O(\log \Delta \log^3 k)$ [14], and the k-taxi problem an $O(\log^3 \Delta \log^2(nk\Delta))$-competitive algorithm [32]. However, since n and Δ can be infinite in general, these bounds provide no improvement even in seemingly simple metrics such as the line metric or $[0, 1]^2$ with some norm.

We argue that smoothed analysis is a realistic model for these problems. For example, a customer's pick-up location in taxi apps is often determined through somewhat inaccurate GPS or by placing a pin on a map by hand, and even if an address is specified this usually does not correspond to an exact point but rather a small area on a map. Even in cases where exact precision in response to arriving requests is important, the way that inputs are generated is rarely fully adversarial and may be subject to noise. If smoothed analysis can drastically improve the competitive ratio of these problems, we believe that algorithms achieving these improved guarantees have a much higher chance of being adopted in practice than pessimistic worst-case methods.

We consider the setting where the metric space M is a bounded diameter subset of some normed space (and therefore contained in some ball B_M). For each time step, an adversary can choose a distribution from which the next request is drawn, subject to the constraint that the density functions of points appearing in the requests are upper bounded by $1/\sigma$ times the density of the uniform distribution over B_M. Thus, as $\sigma \to 0$, this converges to the classical worst-case setting, whereas $\sigma = 1$ means that request locations are drawn uniformly from $M = B_M$. Small values of σ can model, for example, request locations that are slightly perturbed in a small ball around the worst-case locations.

For all three problems, we obtain $\mathrm{polylog}(k/\sigma)$-competitive algorithms in this setting via simple reductions to the adversarial setting in finite metrics. Thus, for σ polynomial in k we obtain an exponential improvement over the known worst-case guarantees for k-server and chasing small sets, and an infinite improvement for the k-taxi problem. Our algorithms require no knowledge about σ (except for the k-taxi problem, where a non-zero lower bound on σ is requied), and beyond these smoothed bounds they still also achieve the known worst-case guarantees (where they exist, i.e., for k-server and chasing small sets).

Further Related Work. The k-server problem and an easier version of the k-taxi problem was previously studied under a stochastic setting different from smoothed analysis by Dehgani et al. [21]. They consider the case where each request is drawn independently from some distribution, and these distributions are known to the algorithm. For this setting, they give an approximation algorithm for the line and finite metrics that is competitive against the best *online* algorithm for the instance. In contrast, we do not assume knowledge of the distributions, and compare against the best offline algorithm for the input.

Several online problems have been studied in a stochastic setting where requests are drawn i.i.d. from some distribution, see e.g. [28, 30, 31]. This differs from our setting, as we do not require distributions to be identical or independent.

Another beyond-worst-case model that has received significant interest in recent years is the learning-augmented paradigm [39, 42], where an algorithm's input is augmented with additional predictions about future input or best behavior.

The k-server problem, introduced in [40], is considered one of the most fundamental problems in online algorithms and is often called the "holy grail of competitive analysis". For deterministic algorithms the $\Theta(k)$ bound is known to be tight [40, 38] up to a constant factor, and the k-server conjecture states that it is exactly k. For randomized algorithms, the aforementioned improvements for the special cases of finite metrics and bounded aspect ratios are known [14]. The randomized k-server conjecture, positing an $O(\log k)$-competitive algorithm for general metrics, was recently refuted in [12]. The k-taxi problem is known to be at least exponentially harder than the k-server problem at least for deterministic algorithms, with an $\Omega(2^k)$ lower bound and no known upper bound for general metrics [18]. For chasing small sets, the deterministic competitive ratio is known to lie in $O(k2^k) \cap \Omega(2^k)$ [26, 16], and the randomized competitive ratio is $\Theta(k^2)$ [11, 12].

Chasing small sets is a special case of the metrical task systems (MTS) problem [9]. In MTS, each request consists of a cost function, and the algorithm which moves the single server incurs both the "service cost" as well as the movement cost. Chasing small sets corresponds to MTS with request cost functions restricted to set-membership indicator functions. The k-server problem is also a special case of MTS, though not in the same metric space but rather the metric space of configurations.

1.1 Preliminaries

All the problems we consider are defined over a metric space (M, d). The input revealed online is a sequence of requests $r_1, r_2, \ldots, r_T$, each of which must be served immediately as specified below.

k-Server. The algorithm controls the movement of k servers located at points in M. Each request is a point $r_t \in M$, and must be served by moving one of the servers to r_t. The cost of the algorithm is defined as the total distance travelled by all servers.

k-Taxi. The algorithm controls the movement of k taxis located at points in M. Each request is a pair of points $r_t = (a_t, b_t) \in M^2$, representing the start and destination of a ride request. To serve it, the algorithm must choose a taxi and move it first to a_t and then to b_t. The cost is defined as the total distance of "empty runs" (i.e., distance travelled except for the travels from a_t to b_t).

Chasing Small Sets. The algorithm controls the movement of a single server located in M. Each request is a subset $r_t \subset M$ of cardinality at most k. To serve it, the algorithm must move the server to one of the points in r_t. The cost incurred by the algorithm is the total distance travelled.

Smooth Instances. Throughout this paper, we assume that the metric space M is a subset of $\mathbb{R}^m$ and the metric d is induced by a norm, i.e., $d(x, y) = \|x - y\|$ for all $x, y \in M$, where $\|\cdot\|$ is an arbitrary norm. We assume that M has bounded diameter, and denote by B_M a (closed) ball of smallest radius containing M, and by R_M its radius.

We consider the variant of the problems described above where each request r_t is drawn from some probability distribution D_t. These distributions need not be independent, i.e., the distribution D_t (and whether it exists or the sequence stops earlier) can be chosen by the adversary after the realizations of $r_1, \ldots, r_{t-1}$ have been determined (however, without knowledge of any random choices made by the online algorithm). For the k-server problem, D_t is a distribution over M, for the k-taxi over M^2 and for chasing small sets over M^{k_t} for some $k_t \le k$. We denote by D_{ti} the corresponding marginal distributions, where $i = 1$ for k-server, $i \in \{1, 2\}$ for k-taxi and $i \in [k_t]$ for chasing small sets.

In the absence of any restrictions on the probability distribution, this problem setting is equivalent to the familiar worst-case setting, as the adversary can choose a (degenerate) distribution that places all probability mass on the worst-case request sequence. Thus, we impose the following smoothness assumption:

▶ **Definition 1.** *Let $\sigma \in (0, 1]$. An instance is σ-smooth if for each marginal distribution D_{ti}, the probability of any measurable set $S \subseteq M$ is at most a factor $1/\sigma$ larger than under the uniform distribution over B_M.*

Equivalently, this means that the density function of each marginal distribution is upper bounded by $1/(\sigma \cdot \mathrm{vol}(B_M))$. The case $\sigma = 1$ corresponds to uniformly distributed requests, whereas $\sigma \to 0$ captures the classical worst-case setting.

Competitive Ratio in the Smoothed Model. For a (possibly randomized) algorithm A, we denote its expected cost on an instance I by $\mathrm{cost}(A, I)$. Denote by opt the offline algorithm that achieves the lowest possible cost on any instance. In the classical (worst-case) setting, an online algorithm is said to be c-competitive if

$$\mathrm{cost}(A, I) \le c \cdot \mathrm{cost}(\mathrm{opt}, I) + \alpha, \tag{1}$$

for all possible instances I, where α is a constant independent of I. We say that A is c-competitive on σ-smooth instances if inequality (1) holds when I is sampled as a σ-smooth instance and both sides of the inequality are replaced by their expectation over the randomness in I. We allow opt to serve the actual realization of the instance optimally, i.e., knowing the outcome of the random sampling process of I in advance. Of course, any competitive algorithm in this setting is also competitive if opt has to make its choices before knowing the outcomes of the future sampling process of I.

We will usually drop I from the notation when it is understood from the context.

η-Nets. An *η-net* of a metric space M is a subset $N \subseteq M$ satisfying the following properties:

- N is η-dense in M: For all $x \in M$ there exists $y \in N$ such that $d(x, y) \leq \eta$.
- N is η-separated: For all $x, y \in N$ it holds that $d(x, y) > \eta$.

Aspect Ratio. The aspect ratio of a metric space is the ratio between the largest and smallest non-zero distance.

1.2 Our Results

We obtain polylog(k/σ)-competitive algorithms for all three problems we consider, as summarized in the following theorems. By known algorithm combination techniques, our algorithms for k-server and chasing small sets still also achieve the same robustness guarantees as classical worst-case algorithms.

▶ **Theorem 2** (k-Server). *There exists an $O(\min\{k, \log(k/\sigma)\cdot\log^2 k\})$-competitive randomized algorithm for k-server on σ-smooth instances, even if σ is unknown.*

▶ **Theorem 3** (k-Taxi). *There exists an $O(\log^5(k/\sigma))$-competitive randomized algorithm for k-taxi on σ-smooth instances, provided a non-zero lower bound on σ is known.*

▶ **Theorem 4** (Chasing Small Sets). *There exists an $O(\min\{k^2, \log^2(k/\sigma)\})$-competitive randomized algorithm for chasing small sets on σ-smooth instances, even if σ is unknown.*

We complement our upper bounds with the following lower bound, ruling out algorithms with a sub-polylogarithmic dependence on k/σ.

▶ **Theorem 5** (Lower Bound). *For any $c < 1/2$, there is no $O(\log^c(k/\sigma))$-competitive randomized algorithm for general σ-smooth instances of k-server, k-taxi, or chasing small sets when the metric space is a ball in $O(\log k)$-dimensional ℓ_∞-space, even if all distributions D_t are identical and independent.*

1.3 Organization

All our algorithms are based on the same framework, as explained in the next section. The upper bound proofs are completed by separate arguments for bounding the offline cost for the three problems in Section 3. The polylogarithmic lower bound (Theorem 5) is proved in Section 4.

2 Basic Construction

To describe our algorithm, let us assume initially that σ is known. Later we will show how to drop this assumption.

For some suitable $\eta > 0$, we choose a finite η-net N of the metric space M. For each $x \in M$, we pick some point $y \in N$ satisfying $d(x, y) \leq \eta$ and call it the *projection of x onto N*, denoted by $\pi(x) = y$ (making an arbitrary choice in case there are several options). Similarly, for a request r_t we denote by $\pi(r_t)$ the projection of r_t, where each point in r_t is replaced by its projection. As the request sequence $r_1, r_2, \ldots, r_T$ in M is revealed, we simulate some online algorithm A_N that operates on finite metrics to serve the request sequence $\pi(r_1), \pi(r_2), \ldots, \pi(r_T)$ in metric space (N, d). Our online algorithm A_M for the original problem on M follows the configurations of A_N at all times, except for an additional movement from $\pi(r_t)$ to r_t and back at time t, to make sure A_M serves the original request sequence. (For example, for a taxi request (a_t, b_t), algorithm A_M first moves the same taxi as A_N to the point $\pi(a_t)$, then pays at most η to move from $\pi(a_t)$ to a_t; from here, it serves the ride request, which changes the taxi's location from a_t to b_t (at no cost); finally, it moves from b_t to $\pi(b_t)$ for another cost of at most η, restoring the invariant that A_M is in the same configuration as A_N.) Thus, the cost of A_M exceeds the cost A_N by at most 2η per request, and the total cost of A_M is at most $2\eta \cdot T + \eta \cdot k$ larger for a sequence of T requests, with the additional cost ηk for projecting the initial configuration onto N. (In the case of chasing small sets, the $\eta \cdot k$ term can also be replaced by η.)

2.1 Reduction to Worst-Case Setting

Our analysis is based on the observation that the smoothing model forces the offline optimal algorithm to incur a sufficiently large movement cost. In particular, we have:

▶ **Proposition 6.** *For σ-smooth instances of k-server, k-taxi, and chasing small sets, the offline optimal cost is at least $\Omega\left(R_M \cdot \left(\frac{\sigma}{\mathrm{poly}(k)}\right)^{1/m}\right)$ amortized per request in expectation, where $\mathrm{poly}(k) = O(k)$ for k-server and k-taxi and $\mathrm{poly}(k) = O(k^2)$ for chasing small sets.*

More precise statements for the respective problems along with a proof of Proposition 6 are given in Section 3.

If η is within a constant factor of the bound in Proposition 6 and algorithm A_N is c_N-competitive for the worst-case (non-smoothed) problem in N, then we can show that algorithm A_M described above is $O(c_N)$-competitive for the smoothed problem in M:

▶ **Lemma 7.** *If A_N is c_N-competitive in metric space (N, d), then algorithm A_M is $O(c_N)$-competitive on instances in M whose optimal offline cost is at least $\Omega(\eta)$ amortized per request.*

Proof. Denote by opt_M and opt_N, respectively, an optimal offline algorithm for the instance in M and the projected instance in N. By assumption

$$\mathrm{cost}(A_N) \leq c_N \cdot \mathrm{cost}(\mathrm{opt}_N) + \alpha \tag{2}$$

for some constant α. Denote by T the number of requests and by $\mathbb{E}[T]$ its expectation[1]. As argued above,

$$\mathrm{cost}(A_M) \leq \mathrm{cost}(A_N) + 2\eta \cdot \mathbb{E}[T] + \eta \cdot k, \tag{3}$$

[1] T may be random as the adversary can choose the stopping point depending on the realizations of earlier requests.

with the additional cost due to movement between $\pi(r_t)$ and r_t at each time. By a similar argument,

$$\text{cost}(\text{opt}_N) \leq \text{cost}(\text{opt}_M) + 2\eta \cdot \mathbb{E}[T], \tag{4}$$

as a possible offline algorithm to service the requests $\pi(r_1), \pi(r_2), \ldots, \pi(r_T)$ in N is to always be in the configuration of opt_M projected onto N, incurring at most an additional cost 2η per request by the triangle inequality (noting that, without loss of generality, opt_M moves at most one server/taxi per time step). Combining inequalities (2), (3) and (4) gives

$$\text{cost}(A_M) \leq c_N \cdot \text{cost}(\text{opt}_M) + (c_N + 1) \cdot 2\eta \cdot \mathbb{E}[T] + \eta \cdot k + \alpha.$$

Next, we make use of the assumption that the optimal offline cost is at least $\Omega(\eta)$ amortized per request in expectation, i.e., $\text{cost}(\text{opt}_M) \geq \Omega(\eta) \cdot \mathbb{E}[T] - \beta$ for some constant β. Substituting this into the previous inequality gives

$$\text{cost}(A_M) \leq O(c_N) \cdot \text{cost}(\text{opt}_M) + \eta \cdot k + \alpha + O(\beta \cdot c_N),$$

and the lemma follows since the additive term $\eta \cdot k + \alpha + O(\beta \cdot c_N)$ is a constant independent of the request sequence. ◀

2.2 Size of the Net N

Lemma 7 allows us to reduce smoothed instances on M to non-smooth instances on N, provided the offline optimal cost is at least $\Omega(\eta)$ per request. For each of the problems we consider, by known results for finite metric spaces, the competitive ratio c_N can be expressed as a function of the number of points in N and possibly the aspect ratio Δ of N. Specifically, for k-server, the (best known) competitive ratio in n-point metric spaces is $O(\log^2 k \cdot \log n)$ [14, 15], for k-taxi it is $O(\log^3 \Delta \cdot \log^2(nk\Delta))$ [32], and for chasing small sets (which is a special case of metrical task systems) it is $O(\log^2 n)$ [13, 19].

Thus, it remains to choose η (i) large enough to obtain an appropriate value for the cardinality and aspect ratio of N and (ii) small enough to ensure the lower bound in Proposition 6 becomes $\Omega(\eta)$.

The following lemma provides the desired relationship between the parameter η and the number of points in N.

▶ **Lemma 8.** *For each $\eta \in (0, R_M]$, there exists an η-net N of M of size $|N| \leq \left(\frac{3R_M}{\eta}\right)^m$.*

Proof. The proof is similar to [25, Claim 2.6]. We initialize N as a singleton set containing an arbitrary point from M; then as long as there exists a point $x \in M$ such that $d(x, y) > \eta$ for all $y \in N$, we add x to N. By construction, N is an η-net. Moreover, the balls of radius $\eta/2$ centered at the points in N are disjoint and contained in a ball of radius $R_M + \eta/2$ (namely, the ball obtained by extending the radius of B_M by $\eta/2$). The ratio between the volume of a ball of radius $\eta/2$ and the ball of radius $R_M + \eta/2$ is $\left(\frac{\eta/2}{R_M+\eta/2}\right)^m$. Thus, there can be at most $\left(\frac{R_M+\eta/2}{\eta/2}\right)^m \leq \left(\frac{3R_M}{\eta}\right)^m$ disjoint balls. ◀

2.3 Putting it all Together

We now combine the above statements to prove the following statement.

▶ **Proposition 9.** *For known σ, there exists a randomized online algorithm whose competitive ratio on σ-smooth instances is $O(\log(k/\sigma) \cdot \log^2 k)$ for k-server, $O(\log^5(k/\sigma))$ for k-taxi and $O(\log^2(k/\sigma))$ for chasing small sets.*

Proof. Let $\eta = 3R_M \cdot \left(\frac{\sigma}{\text{poly}(k)}\right)^{1/m}$, where $\text{poly}(k)$ is the function from Proposition 6. Then A_M is $O(c_N)$-competitive by Proposition 6 and Lemma 7. If $\eta > R_M$, then N can be chosen as the singleton containing only the center of the ball B_M, which trivially yields $c_N = 1$ and the theorems. Otherwise, by Lemma 8 the size of the η-net N is

$$n := |N| \leq \frac{\text{poly}(k)}{\sigma},$$

and since the diameter of M is at most $2R_M$, the aspect ratio of N is

$$\Delta \leq \frac{2R_M}{\eta} = \frac{2}{3}\left(\frac{\text{poly}(k)}{\sigma}\right)^{1/m}.$$

Thus, since $c_N = O(\log^2 k \cdot \log n)$ for k-server [14, 15], $c_N = O(\log^3 \Delta \cdot \log^2(nk\Delta))$ for k-taxi [32], and $c_N = O(\log^2 n)$ for metrical task system [13, 19], and chasing small sets is a special case of metrical task systems (in the same metric space), the result follows. ◀

We note that the competitive ratio for k-taxi could actually be improved by a factor $1/\min\{m, \log(k/\sigma)\}^3$. (Note that the $\log \Delta$ term stands for a quantity that is at least 1, so the improvement is not greater.) This would typically only yield a constant factor improvement though, considering practical applications of the k-taxi problem usually having a small dimension of at most 3.

2.4 Unknown σ and Robustness

If σ is unknown, and to achieve the robustness bounds in our main theorems, the idea is to simulate in parallel several instances of the above algorithm for different values of σ as well as a worst-case algorithm, and combine them using the technique of Blum and Burch [8].

▶ **Theorem 10** (Blum and Burch [8])**.** *Given $\epsilon > 0$ and ℓ online algorithms $A_1, \ldots, A_\ell$ for a problem that can be formulated as a metrical task system of diameter* diam*, there exists an online algorithm whose expected cost is at most*

$$(1+\epsilon) \cdot \min_{i \in [\ell]} \text{cost}(A_i) + O\left(\frac{\text{diam} \cdot \log \ell}{\epsilon}\right).$$

Chasing Small Sets. Since chasing small sets in a metric space (M, d) is a metrical task system in the same metric space, we can invoke the above theorem with $\epsilon = 1$, $\ell = \lceil \log_2 k \rceil + 1$ and using as A_i for $i = 1, \ldots, \ell - 1$ our algorithm with smoothness parameter $\sigma_i = 2^{-2^i}$, and as A_ℓ the $\Theta(k^2)$-competitive algorithm from [11] for chasing small sets in the fully adversarial setting.

Since $\text{diam} \leq 2R_M$, the additive term in Theorem 10 is a constant independent of the request sequence. Thus, the resulting algorithm achieves, up to a factor $1 + \epsilon \leq 2$, the same competitive ratio as the best of the algorithms A_i. If the true smoothness parameter $\sigma < 2^{-(\ell-1)} \leq 2^{-k}$, then $O(\min\{k^2, \log^2(k/\sigma)\})$-competitiveness follows since the minimum is attained by k^2. Otherwise, there exists i such that $\sigma_i \leq \sigma \leq \sqrt{\sigma_i}$. Since any σ-smooth instance is also σ_i-smooth, we achieve competitive ratio $O(\log^2(k/\sigma_i)) = O(\log^2(k/\sigma))$. This yields Theorem 4.

***k*-Server.** For the k-server problem (Theorem 2), the same argument works since k-server can be modelled as a metrical task system in the space of server configurations (whose diameter is k times the diameter of the original metric space). The only difference is that the algorithm A_ℓ for general metrics in the fully adversarial setting has competitive ratio $\Theta(k)$ [38].

***k*-Taxi.** It is not known how to model the k-taxi problem as a metrical task system, due to configuration changes that incur no cost.[2] Nonetheless, the proof of Theorem 10 in [8] extends to this setting without any change, so we can still apply the same combination method also to the k-taxi problem. However, since no competitive algorithm for the k-taxi problem on general metrics in the fully adversarial setting is known, we require a lower bound on σ to ensure that the number of combined algorithms is finite.

3 Bounding Offline Cost

3.1 *k*-Server

▶ **Lemma 11.** *For σ-smooth instances of the k-server problem, the optimal offline algorithm pays an expected cost of at least*

$$\mathrm{cost}(\mathrm{opt}_M) \geq \frac{R_M}{8} \cdot \left(\frac{\sigma}{8k}\right)^{1/m} \cdot \mathbb{E}[T] - c,$$

where $c > 0$ is a constant independent of the request sequence.

Proof. Let $\delta > 0$. Consider a subsequence $r_{t+1}, r_{t+2}, \ldots, r_{t+4k}$ of $4k$ consecutive requests. For $i = 0, \ldots, 4k$, we denote by S_i a δ-separated subset of $\{r_{t+1}, r_{t+2}, \ldots, r_{t+i}\}$, constructed as follows: $S_0 = \emptyset$, and for $i \geq 1$,

$$S_i = \begin{cases} S_{i-1} \cup \{r_{t+i}\} & \text{if } S_{i-1} \cup \{r_{t+i}\} \text{ is } \delta\text{-separated} \\ S_{i-1} & \text{otherwise.} \end{cases}$$

Note that $S_i = S_{i-1}$ if and only if r_{t+i} belongs to a ball of radius δ centered at a point in S_{i-1}. The volume of each such ball is at most a $\left(\frac{\delta}{R_M}\right)^m$ fraction of the volume of B_M, and thus its probability under D_{t+i} is at most $\left(\frac{\delta}{R_M}\right)^m / \sigma$. Hence, since there are at most $4k$ such balls of radius δ, the probability that $S_i = S_{i-1}$ is bounded as

$$P\left(S_i = S_{i-1}\right) \leq \frac{4k}{\sigma} \left(\frac{\delta}{R_M}\right)^m.$$

We now choose $\delta = R_M \cdot \left(\frac{\sigma}{8k}\right)^{1/m}$, so that the right hand side becomes $1/2$. In other words, in each step, another point is added to S_i with probability at least $1/2$. Thus, after $4k$ steps we have $|S_{4k}| \geq 2k$ with probability at least $1/2$. If this is the case, then any offline algorithm must pay at least $k \cdot \delta$ to serve these requests, regardless of the configuration it starts at: Indeed, at most k of the requests in S_{4k} can be served by a server that previously served no other request from S_{4k}. For the remaining $|S_{4k}| - k \geq k$ requests, the serving server must have moved at least distance δ since the previous time it served a request from S_{4k}, since

[2] Though it belongs to the more general class of metrical task systems with transformation [10].

S_{4k} is δ-separated. Thus, any subsequence of $4k$ consecutive requests contributes $k \cdot \delta/2$ to the expected optimal cost. Over a sequence of T requests, this leads to an expected optimal cost of

$$\begin{aligned}\operatorname{cost}(\operatorname{opt}_M) &\geq \mathbb{E}\left[\left\lfloor \frac{T}{4k} \right\rfloor\right] \cdot \frac{k \cdot \delta}{2} \\ &\geq \frac{\mathbb{E}[T] \cdot \delta}{8} - \frac{k \cdot \delta}{2} \\ &= \frac{\mathbb{E}[T] \cdot R_M \cdot \left(\frac{\sigma}{8k}\right)^{1/m}}{8} - \frac{k \cdot R_M \cdot \left(\frac{\sigma}{8k}\right)^{1/m}}{2}\end{aligned}$$

◀

3.2 k-Taxi

▶ **Lemma 12.** *For σ-smooth instances of the k-taxi problem, the optimal offline algorithm pays an expected cost of at least*

$$\operatorname{cost}(\operatorname{opt}_M) \geq \frac{R_M}{8} \cdot \left(\frac{\sigma}{8k}\right)^{1/m} \cdot \mathbb{E}[T] - c,$$

where $c > 0$ is a constant independent of the request sequence.

Proof. We proceed similarly to the proof of Lemma 11, except in the construction of S_i we add a request $r_{t+i} = (a_{t+i}, b_{t+i})$ if its start location a_{t+i} is more than distance δ from the destinations of *all* the requests $r_{t+1}, \ldots, r_{t+i-1}$:

$$S_i = \begin{cases} S_{i-1} \cup \{r_{t+i}\} & \text{if } d(b_{t+j}, a_{t+i}) > \delta \text{ for all } j = 1, \ldots, i-1 \\ S_{i-1} & \text{otherwise.} \end{cases}$$

Again,

$$P\left(S_i = S_{i-1}\right) \leq \frac{4k}{\sigma}\left(\frac{\delta}{R_M}\right)^m$$

as $S_i = S_{i-1}$ requires a_{t+i} to fall in one of at most $4k$ balls of radius δ. As before, the same choice of δ yields $|S_{4k}| \geq 2k$ with probability at least $1/2$. If this is the case, then any offline algorithm must pay at least $k \cdot \delta$ to serve these requests: At least $|S_{4k}| - k \geq k$ of the requests in S_{4k} are served by a taxi that also served a previous request since time $t+1$, incurring cost at least δ by definition of S_{4k}. The lemma is then concluded in an identical fashion to the proof of Lemma 11. ◀

3.3 Chasing Small Sets

▶ **Lemma 13.** *In σ-smoothed instances of chasing small sets, the optimal offline algorithm pays an expected cost of at least*

$$\operatorname{cost}(\operatorname{opt}_M) \geq \frac{R_M}{2} \cdot \left(\frac{\sigma}{2k^2}\right)^{1/m} \cdot \mathbb{E}[T].$$

Proof. Let $\delta > 0$. Consider two consecutive request sets r_{t-1} and r_t. The probability that the ith point in r_{t-1} is within distance δ from the jth point in r_t is at most $\frac{1}{\sigma}\left(\frac{\delta}{R_M}\right)^m$. Thus, by a union bound over the at most k^2 pairs of points in $r_{t-1} \times r_t$, the probability that some point in r_{t-1} is within distance δ from some point in r_t is at most $\frac{k^2}{\sigma}\left(\frac{\delta}{R_M}\right)^m$. Otherwise, with probability at least $1 - \frac{k^2}{\sigma}\left(\frac{\delta}{R_M}\right)^m$, any offline algorithm must pay at least δ at time t to move from r_{t-1} to r_t. Choosing $\delta = R_M \cdot \left(\frac{\sigma}{2k^2}\right)^{1/m}$, we see that the probability is at least $1/2$. Thus, the expected offline cost is at least $\frac{\delta}{2} = \frac{R_M}{2} \cdot \left(\frac{\sigma}{2k^2}\right)^{1/m}$ per time step. ◀

4 Lower Bound

Complementing our upper bounds, we now prove a polylog(k/σ) lower bound for the three problems, i.e., that no algorithm can achieve a competitive ratio of $O(\log^c(k/\sigma))$ for any $c < 1/2$ (Theorem 5).

Our proof is based on the classical $\Omega(\log k)$ lower bound for k-server and MTS on $(k+1)$-point uniform metrics, in which each request location is drawn uniformly at random from the $k+1$ points [9]. As σ-smooth instances require distributions with infinite support, we intend to simulate this by substituting each of the $k+1$ points by a small neighborhood in normed space. We describe the construction for k-server first, and then explain how to adapt it to k-taxi and chasing small sets.

4.1 k-Server

We first recall the $\Omega(\log k)$ lower bound from [9]:[3] The underlying metric space is a uniform metric of $k+1$ points (i.e., all pairwise distances are 1) and each request is drawn uniformly at random from the $k+1$ points. Thus, an online algorithm must move at each time step with probability $1/(k+1)$, paying $T/(k+1)$ in expectation for a request sequence of length T. In contrast, an offline algorithm can, whenever a request arrives at the (unique) location not covered by a server, take the server from the location whose next request is furthest in the future. Thus, it need not move again until each of the k other locations is requested again at least once. By a standard coupon collector argument, it takes $\frac{k+1}{k} + \frac{k+1}{k-1} + \cdots + \frac{k+1}{1} = \Theta(k \log k)$ steps in expectation for each of the k other locations to be requested again, so the algorithm only needs to move roughly once every $\Theta(k \log k)$ steps. For a sequence of T requests, this leads to an optimal offline cost of $O\left(\frac{T}{k \log k} + 1\right)$ (see [9] for details). The ratio between the online cost $T/(k+1)$ and offline cost $O\left(\frac{T}{k \log k} + 1\right)$ yields the known $\Omega(\log k)$ lower bound for uniform metrics.

To simulate this in the smoothed setting, let $m = \lceil \log_2(k+1) \rceil$ and take as metric space the hypercube $[0,1]^m$ with ℓ_∞ distance, which is a ball in ℓ_∞^m. Let V be a fixed set of some $k+1$ vertices of this hypercube (i.e., with integer coordinates). Let $\epsilon = \frac{1}{2k \log_2 k} \leq \frac{1}{4}$. We say two points are *near* each other if their distance is at most ϵ. Otherwise, they are *far*. Let P be the set of points in $[0,1]^m$ that are near some vertex in V. Consider the instance where each request is drawn uniformly at random from P.

For the online algorithm, each request incurs cost at least $\Omega(1/k)$ in expectation, since with probability at least $\frac{1}{k+1}$ the request arrives near a vertex v whereas all online algorithm servers are near vertices in $V \setminus \{v\}$. For the offline algorithm, if each request arrived at the nearby vertex in V instead of its actual location in P, then the instance would be precisely the aforementioned instance on a $(k+1)$-point uniform metric, with an offline cost of $O\left(\frac{T}{k \log k} + 1\right)$ for T requests. The fact requests arrive *near* vertices instead of *at* vertices only increases the offline cost by at most ϵ per request, as the distance between any two points near the same vertex is at most ϵ. By choice of ϵ, this still leads to an offline cost of $O\left(\frac{T}{k \log k} + 1\right)$, and a competitive ratio lower bound of $\Omega(\log k)$.

It remains to express the lower bound in terms of k/σ. The overall hypercube $[0,1]^m$ has a volume of 1, whereas the set P has a volume of $(k+1)\epsilon^m$. Thus, $\sigma = \frac{1}{(k+1)\epsilon^m} = 2^{-\Theta(\log^2 k)}$, and $k/\sigma = 2^{\Theta(\log^2 k)}$. Hence, the lower bound can be written as $\Omega(\log^{1/2}(k/\sigma))$.

[3] The presentation in [9] is for MTS. We describe it here in the language of k-server.

4.2 k-Taxi

The construction is similar, but requests need to be pairs of two points (a, b). For each request, we sample a as above, and if a is near a vertex v, then we sample b uniformly at random from the points near v as well. This ensures that an offline algorithm can keep its taxis near k distinct vertices. The only difference in calculations is that σ is larger by a factor $k + 1$ as request destinations are sampled from distributions supported on points near a single vertex, but the asymptotic bound $\sigma = 2^{-\Theta(\log^2 k)}$ is unaffected by this.

4.3 Chasing Small Sets

We again use a similar construction. Each request set contains k points in P, sampled uniformly at random subject to the constraint that the points are near k distinct vertices of V. Thus, for each request set there is one vertex in V that is far from it. The online cost is still $\Omega(1/k)$ per request in expectation, as with probability at least $1/(k + 1)$ the request set contains no point near the algorithm's server. The offline cost is also bounded as before, as whenever it is far from all points in the request set, it can first move to the vertex for which it will happen furthest in the future that it is far from all points in the request set; additionally, it pays at most ϵ per request to move to a nearby point in the request set. The lower bound $\Omega(\log^{1/2}(k/\sigma))$ follows in the same way.

References

1. Omer Angel, Sébastien Bubeck, Yuval Peres, and Fan Wei. Local max-cut in smoothed polynomial time. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, 2017.
2. David Arthur, Bodo Manthey, and Heiko Röglin. Smoothed analysis of the k-means method. *J. ACM*, 58(5), 2011. doi:10.1145/2027216.2027217.
3. David Arthur and Sergei Vassilvitskii. How slow is the k-means method? In Nina Amenta and Otfried Cheong, editors, *Proceedings of the 22nd ACM Symposium on Computational Geometry*, 2006.
4. Nikhil Bansal, Haotian Jiang, Raghu Meka, Sahil Singla, and Makrand Sinha. Prefix discrepancy, smoothed analysis, and combinatorial vector balancing. In *13th Innovations in Theoretical Computer Science Conference, ITCS*, 2022. doi:10.4230/LIPIcs.ITCS.2022.13.
5. Nikhil Bansal, Haotian Jiang, Raghu Meka, Sahil Singla, and Makrand Sinha. Smoothed analysis of the komlós conjecture. In *49th International Colloquium on Automata, Languages, and Programming, ICALP*, 2022. doi:10.4230/LIPIcs.ICALP.2022.14.
6. Luca Becchetti, Stefano Leonardi, Alberto Marchetti-Spaccamela, Guido Schäfer, and Tjark Vredeveld. Average-case and smoothed competitive analysis of the multilevel feedback algorithm. *Mathematics of Operations Research*, 31(1):85–108, 2006. doi:10.1287/MOOR.1050.0170.
7. Adam Block, Yuval Dagan, Noah Golowich, and Alexander Rakhlin. Smoothed online learning is as easy as statistical learning. In *Conference on Learning Theory, COLT*, 2022.
8. Avrim Blum and Carl Burch. On-line learning and the metrical task system problem. *Mach. Learn.*, 39(1), 2000. doi:10.1023/A:1007621832648.
9. Allan Borodin, Nathan Linial, and Michael E Saks. An optimal on-line algorithm for metrical task system. *Journal of the ACM (JACM)*, 39(4):745–763, 1992. doi:10.1145/146585.146588.
10. Sébastien Bubeck, Niv Buchbinder, Christian Coester, and Mark Sellke. Metrical service systems with transformations. In *12th Innovations in Theoretical Computer Science Conference, ITCS*, 2021. doi:10.4230/LIPIcs.ITCS.2021.21.
11. Sébastien Bubeck, Christian Coester, and Yuval Rabani. Shortest paths without a map, but with an entropic regularizer. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS*, 2022.

12 Sébastien Bubeck, Christian Coester, and Yuval Rabani. The randomized k-server conjecture is false! In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 581–594, 2023. doi:10.1145/3564246.3585132.

13 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*, 50(3):909–923, 2021. doi:10.1137/19M1237879.

14 Sébastien Bubeck, Michael B Cohen, Yin Tat Lee, James R Lee, and Aleksander Madry. K-server via multiscale entropic regularization. In *Proceedings of the 50th annual ACM SIGACT symposium on theory of computing*, pages 3–16, 2018. doi:10.1145/3188745.3188798.

15 Niv Buchbinder, Anupam Gupta, Marco Molinaro, and Joseph (Seffi) Naor. k-servers with a smile: Online algorithms via projections. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, 2019.

16 William R. Burley. Traversing layered graphs using the work function algorithm. *J. Algorithms*, 20(3), 1996. doi:10.1006/JAGM.1996.0024.

17 Marek Chrobak and Lawrence L. Larmore. The server problem and on-line games. In *On-Line Algorithms, Proceedings of a DIMACS Workshop*. DIMACS/AMS, 1991. doi:10.1090/DIMACS/007/02.

18 Christian Coester and Elias Koutsoupias. The online k-taxi problem. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC*, 2019.

19 Christian Coester and James R. Lee. Pure entropic regularization for metrical task systems. *Theory Comput.*, 18:1–24, 2022. doi:10.4086/TOC.2022.V018A023.

20 Vincent Cohen-Addad and Varun Kanade. Online optimization of smoothed piecewise constant functions. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS*, 2017.

21 Sina Dehghani, Soheil Ehsani, MohammadTaghi Hajiaghayi, Vahid Liaghat, and Saeed Seddighin. Stochastic k-server: How should uber work? In *44th International Colloquium on Automata, Languages, and Programming, ICALP*, 2017.

22 Naveen Durvasula, Nika Haghtalab, and Manolis Zampetakis. Smoothed analysis of online non-parametric auctions. In *Proceedings of the 24th ACM Conference on Economics and Computation, EC*, 2023.

23 Matthias Englert, Heiko Röglin, and Berthold Vöcking. Worst case and probabilistic analysis of the 2-opt algorithm for the TSP. *Algorithmica*, 68(1), 2014. doi:10.1007/S00453-013-9801-4.

24 Matthias Englert, Heiko Röglin, and Berthold Vöcking. Smoothed analysis of the 2-opt algorithm for the general TSP. *ACM Trans. Algorithms*, 13(1), 2016. doi:10.1145/2972953.

25 Nova Fandina and Yair Bartal. Lecture notes in Metric embedding theory and its algorithmic applications, 2018. Accessed on June 29, 2024. URL: https://moodle2.cs.huji.ac.il/nu18/pluginfile.php/274639/mod_resource/content/1/METAP19_Lecture_2.pdf.

26 Amos Fiat, Dean P. Foster, Howard J. Karloff, Yuval Rabani, Yiftach Ravid, and Sundar Vishwanathan. Competitive algorithms for layered graph traversal. *SIAM J. Comput.*, 28(2), 1998. doi:10.1137/S0097539795279943.

27 Amos Fiat, Yuval Rabani, and Yiftach Ravid. Competitive k-server algorithms (extended abstract). In *31st Annual Symposium on Foundations of Computer Science, FOCS*, 1990.

28 Naveen Garg, Anupam Gupta, Stefano Leonardi, and Piotr Sankowski. Stochastic analyses for online combinatorial optimization problems. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, 2008.

29 Yiannis Giannakopoulos, Alexander Grosz, and Themistoklis Melissourgos. On the smoothed complexity of combinatorial local search. In *51st International Colloquium on Automata, Languages, and Programming, ICALP*, 2024. doi:10.4230/LIPIcs.ICALP.2024.72.

30 Fabrizio Grandoni, Anupam Gupta, Stefano Leonardi, Pauli Miettinen, Piotr Sankowski, and Mohit Singh. Set covering with our eyes closed. *SIAM J. Comput.*, 42(3):808–830, 2013. doi:10.1137/100802888.

31 Anupam Gupta, Guru Guruganesh, Binghui Peng, and David Wajc. Stochastic online metric matching. In *46th International Colloquium on Automata, Languages, and Programming, ICALP*, 2019. doi:10.4230/LIPIcs.ICALP.2019.67.

32 Anupam Gupta, Amit Kumar, and Debmalya Panigrahi. Poly-logarithmic competitiveness for the k-taxi problem. In *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA*, 2024.

33 Rishi Gupta and Tim Roughgarden. A PAC approach to application-specific algorithm selection. *SIAM J. Comput.*, 46(3), 2017. doi:10.1137/15M1050276.

34 Nika Haghtalab, Yanjun Han, Abhishek Shetty, and Kunhe Yang. Oracle-efficient online learning for smoothed adversaries. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS*, 2022.

35 Nika Haghtalab, Tim Roughgarden, and Abhishek Shetty. Smoothed analysis of online and differentially private learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS*, 2020.

36 Nika Haghtalab, Tim Roughgarden, and Abhishek Shetty. Smoothed analysis with adaptive adversaries. *J. ACM*, 71(3), 2024. doi:10.1145/3656638.

37 Sampath Kannan, Jamie Morgenstern, Aaron Roth, Bo Waggoner, and Zhiwei Steven Wu. A smoothed analysis of the greedy algorithm for the linear contextual bandit problem. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS*, 2018.

38 Elias Koutsoupias and Christos H. Papadimitriou. On the k-server conjecture. *J. ACM*, 42(5), 1995. doi:10.1145/210118.210128.

39 Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. *J. ACM*, 68(4), 2021. doi:10.1145/3447579.

40 Mark S. Manasse, Lyle A. McGeoch, and Daniel Dominic Sleator. Competitive algorithms for server problems. *J. Algorithms*, 11(2), 1990. doi:10.1016/0196-6774(90)90003-W.

41 Bodo Manthey and Rianne Veenstra. Smoothed analysis of the 2-opt heuristic for the TSP: polynomial bounds for gaussian noise. In *Algorithms and Computation - 24th International Symposium, ISAAC*, 2013.

42 Michael Mitzenmacher and Sergei Vassilvitskii. Algorithms with predictions. *Commun. ACM*, 65(7), 2022. doi:10.1145/3528087.

43 Christos H Papadimitriou and Mihalis Yannakakis. Shortest paths without a map. *Theoretical Computer Science*, 84(1):127–150, 1991. doi:10.1016/0304-3975(91)90263-2.

44 Manish Raghavan, Aleksandrs Slivkins, Jennifer Wortman Vaughan, and Zhiwei Steven Wu. The externalities of exploration and how data diversity helps exploitation. In *Conference On Learning Theory, COLT*, 2018.

45 Alexander Rakhlin, Karthik Sridharan, and Ambuj Tewari. Online learning: Stochastic, constrained, and smoothed adversaries. *Advances in neural information processing systems*, 2011.

46 Tim Roughgarden, editor. *Beyond the Worst-Case Analysis of Algorithms*. Cambridge University Press, 2020.

47 Guido Schäfer and Naveen Sivadasan. Topology matters: Smoothed competitiveness of metrical task systems. *Theoretical Computer Science*, 341(1-3):216–246, 2005. doi:10.1016/J.TCS.2005.04.006.

48 Daniel A Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM (JACM)*, 51(3):385–463, 2004. doi:10.1145/990308.990310.