Tracking Moving Objects with Few Handovers

David Eppstein, Michael T. Goodrich, and Maarten Löffler

Dept. of Computer Science, Univ. of California, Irvine

Abstract. We study the online problem of assigning a moving point to a base-station region that contains it. Our goal is to minimize the number of *handovers* that occur when the point moves outside its assigned region and must be assigned to a new one. We study this problem in terms of a competitive analysis measured as a function of Δ, the *ply* of the system of regions, that is, the maximum number of regions that cover any single point.

1 Introduction

A common problem in wireless sensor networks involves the online tracking of moving objects [6, 9, 14, 20, 21]. Whenever a moving object leaves a region corresponding to its tracking sensor, a nearby sensor must take over the job of tracking the object. Similar *handovers* are also used in cellular phone services to track moving customers [16]. In both the sensor tracking and cellular phone applications, handovers involve considerable overhead [6, 9, 14, 16, 21], so we would like to minimize their number.

Geometrically, we can abstract the problem in terms of a set of n closed regions in $\mathbb{R}^d$, for a constant d, which represent the sensors or cell towers. We assume that any pair of regions intersects at most a constant number of times, as would be the case, say, if they were unit disks (a common geometric approximation used for wireless sensors [6, 9, 14, 21]). We also have one or more moving entities, which are represented as points traveling along 1-dimensional curves (which we do not assume to be smooth, algebraic, or otherwise well-behaved, and which may not be known or predictable by our algorithms) with a time stamp associated to each point on the curve (Figure 1).

We need to track the entities via regions that respectively contain them; hence, for each moment in time, we must assign one of the regions to each entity, p, with the requirement that p is inside its assigned region at each moment in time. Finally, we

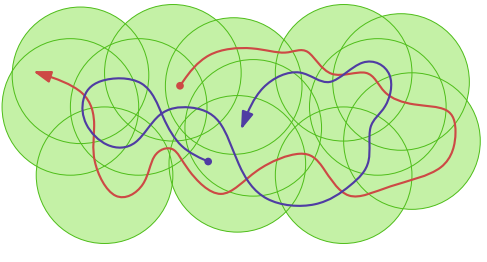


Fig. 1. Example input

F. Dehne, J. Iacono, and J.-R. Sack (Eds.): WADS 2011, LNCS 6844, pp. 362–373, 2011.

want to minimize the number of times that we must change the assignment of the region tracking an entity, so as to minimize the number of handovers.

We also consider a generalized version of this problem, where each point p is required to be assigned to c regions at each moment in time. This generalization is motivated by the need for *trilateration* in cellular networks and wireless sensor networks (e.g., see [18]), where directional information from three or more sensors is used to identify the coordinates of a moving point.

Related Work. There has been considerable previous work in the wireless sensor literature on mobile object tracking. So, rather than providing a complete review of this area, let us simply highlight some of the most relevant work from the wireless sensor literature.

Zhou *et al.* [21] introduce the idea of using handovers to reduce energy in mobile object tracking problems among wireless sensor networks. Pattem *et al.* [14] study energy-quality trade-offs for various strategies of mobile object tracking, including one with explicit handovers. He and Hou [9] likewise study mobile object tracking with respect to handover minimization, deriving probabilistic upper and lower bounds based on distribution assumptions about the moving objects and wireless sensors. Ghica *et al.* [6] study the problem of tracking an object among sensors modeled as unit disks so as to minimize handovers, using probabilistic assumptions about the object's future location while simplifying the tracking requirements to discrete epochs of time.

The analysis tool with which we characterize the performance of our algorithms comes from research in *online algorithms*, where problems are defined in terms of a sequence of decisions that must be made one at a time, before knowing the sequence of future requests. In *competitive analysis* [15], one analyzes an online algorithm by comparing its performance against that of an idealized adversary, who can operate in an offline fashion, making his choices after seeing the entire sequence of items.

We are not aware of any previous work that applies competitive analysis to the problem of handover minimization. Nevertheless, this problem can be viewed from a computational geometry perspective as an instantiation of the *observer-builder* framework of Cho *et al.* [2], which itself is related to the *incremental motion* model of Mount *et al.* [12], the *observer-tracker* model of Yi and Zhang [20], and the well-studied *kinetic data structures* framework [8, 7]. In terms of the observer-builder model, our problem has an *observer* who watches the motion of the point(s) we wish to track and a *builder* who maintains the assignment of tracking region(s) to the point(s). This assignment would define a set of Boolean *certificates*, which become *violated* when a point leaves its currently-assigned tracking region. The observer would notify the builder of any violation, and the builder would use information about the current and past states of the point(s) to make a new assignment (and define an associated certificate). The goal, as in the previous work by Cho *et al.* [2], would be to minimize the number of interactions between the observer and builder, as measured using competitive analysis. Whereas Cho *et al.* apply their model to the maintenance of net trees for moving points, in our case the interactions to be minimized correspond to handovers, and our results supply the algorithms that would be needed to implement a builder for handover minimization. Yi and Zhang [20] study a general online tracking problem, but with a different objective function than ours: when applied to mobile object tracking, rather than optimizing

the number of handovers, their scheme would aim to minimize the distance between objects and the base-station region to which they are each assigned.

Several previous papers study overlap and connectivity problems for geometric regions, often in terms of their *ply*, the maximum number of regions that cover any point. Guibas *et al.* [7] study the maintenance of connectivity information among moving unit disks in the kinetic data structure framework. Miller *et al.* [11] introduce the concept of ply and show how sets of disks with low ply possess small geometric separators. Eppstein *et al.* [3,5] study road network properties and algorithms using a model based on sets of disks with low ply after outliers are removed. Van Leeuwen [17] studies the minimum vertex cover problem for disk graphs, providing an asymptotic FPTAS for this problem on disk graphs of bounded ply. Alon and Smorodinsky [1] likewise study coloring problems for sets of disks with low ply.

Our problem can also be modeled as a *metrical task system* in which the sensor regions are represented as states of the system, the cost of changing from state to state is uniform, and the cost of serving a request is zero for a region that contains the request point and two for other regions. Known randomized online algorithms for metrical task systems [10] would give a competitive ratio of $O(\log n)$ for our problem, not as good as our $O(\log \Delta)$ result, and known lower bounds for metrical task systems would not necessarily apply to our problem.

New Results. In this paper, we study the problem of assigning moving points in the plane to containing base station regions in an online setting and use the competitive analysis to characterize the performance of our algorithms. Our optimization goal in these algorithms is to minimize the number of *handovers* that occur when an object moves outside the range of its currently-assigned base station and must be assigned to a new base station. We measure the competitive ratio of our algorithms as a function of Δ, the *ply* of the system of base station regions, that is, the maximum number of such regions that cover any single point. When object motions are known in advance, as in the offline version of the object traking problem, a simple greedy strategy suffices to determine an optimal assignment of objects to base stations, with as few handovers as possible. For the online problem, on the other hand, for moving points in one dimension, we present a deterministic online algorithm that achieves a competitive ratio of $O(\log \Delta)$, with respect to the offline optimal algorithm, and we show that no better ratio is possible. For two or more dimensions, we present a randomized algorithm that achieves a competitive ratio of $O(\log \Delta)$, and a deterministic algorithm that achieves a competitive ratio of $O(\Delta)$; again, we show that no better ratio is possible.

2 Problem Statement and Notation

Let $\mathscr{D}$ be a set of n regions in $\mathbb{R}^d$. These regions represent the areas that can be covered by a single sensor. We assume that each region is a closed, connected subset of $\mathbb{R}^d$ and that the boundaries of any two regions intersect $O(1)$ times – for instance, this is true when each region is bounded by a piecewise algebraic curve in $\mathbb{R}^2$ with bounded degree and a bounded number of pieces. With these assumptions, the arrangement of the pieces has polynomial complexity $O(n^d)$. The *ply* of $\mathscr{D}$ is defined to be the maximum over $\mathbb{R}^d$

of the number of regions covering any point. We always assume that $\mathscr{D}$ is fixed and known in advance.

Let T be the trajectory of a moving point in $\mathbb{R}^d$. We assume that T is represented as a continuous and piecewise algebraic function from $[0,\infty)$ to $\mathbb{R}^d$, with a finite but possibly large number of pieces. We also assume that each piece of T crosses each region boundary $O(1)$ times and that it is possible to compute these crossing points efficiently. We also assume that $T([0,\infty)) \subset \cup\mathscr{D}$; that is, that the moving point is always within range of at least one sensor; this assumption is not realistic, and we make it only for convenience of exposition. Allowing the point to leave and re-enter $\subset \cup\mathscr{D}$ would not change our results since the handovers caused by these events would be the same for any online algorithm and therefore cannot affect the competitive ratio.

As output, we wish to report a *tracking sequence* S: a sequence of pairs (τ_i, D_i) of a time τ_i on the trajectory (with $\tau_0 = 0$) and a region $D_i \in \mathscr{D}$ that covers the portion of the trajectory from time τ_i to τ_{i+1}. We require that for all i, $\tau_i < \tau_{i+1}$. In addition, for all i, it must be the case that $T([\tau_i, \tau_{i+1}]) \subseteq D_i$, and there should be no $\tau' > \tau_{i+1}$ for which $T([\tau_i, \tau']) \subset D_i$; in other words, once a sensor begins tracking the moving point, it continues tracking that point until it moves out of range and another sensor must take over. Our goal is to minimize $|S|$, the number of pairs in the tracking sequence. We call this number of pairs the *cost* of S; we are interested in finding tracking sequences of small cost.

Our algorithm may not know the trajectory T completely in advance. In the *offline tracking problem*, T is given as input, and we must find the tracking sequence S that minimizes $|S|$; as we show, a simple greedy algorithm accomplishes this task. In the *online tracking problem*, T is given as a sequence of *updates*, each of which specifies a single piece in a piecewise algebraic decomposition of the trajectory T. The algorithm must maintain a tracking sequence S that covers the portion of T that is known so far, and after each update it must extend S by adding additional pairs to it, without changing the pairs that have already been included. As has become standard for situations such as this one in which an online algorithm must make decisions without knowledge of the future, we measure the quality of an algorithm by its *competitive ratio*. Specifically, if a deterministic online algorithm A produces tracking sequence $S_A(T)$ from trajectory T, and the optimal tracking sequence is $S^*(T)$, then the competitive ratio of A (for a given fixed set $\mathscr{D}$ of regions) is

$$\sup_T \frac{|S_A(T)|}{|S^*(T)|}.$$

In the case of a randomized online algorithm, we measure the competitive ratio similarly, using the expected cost of the tracking sequence it generates. In this case, the competitive ratio is

$$\sup_T \frac{E[|S_A(T)|]}{|S^*(T)|}.$$

As a variation of this problem, stemming from trilateration problems in cellular phone network and sensor network coverage, we also consider the problem of finding tracking sequences with *coverage* c. In this setting, we need to report a set of c tracking sequences $S_1, S_2, \ldots, S_c$ for T that are *mutually disjoint* at any point in time: if a region D appears for a time interval $[\tau_i, \tau_{i+1}]$ in one sequence S_k and a time interval $[\sigma_j, \sigma_{j+1}]$

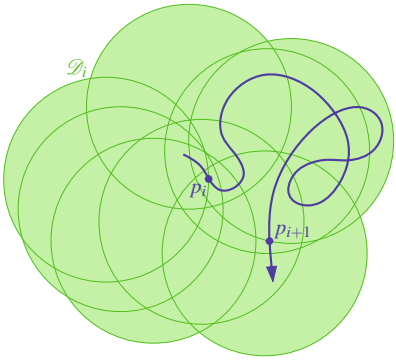


Fig. 2. The set $\mathscr{D}_i$ of disks containing p_i, and the point p_{i+1} where the trajectory leaves the last disk of $\mathscr{D}_i$

in some other sequence S_l, we require that the intervals $[\tau_i, \tau_{i+1}]$ and $[\sigma_j, \sigma_{j+1}]$ are disjoint. We wish to minimize the total cost $\sum_{i=1}^{c} |S_i|$ of a set of tracking sequences with coverage c, and in both the offline and online versions of the problem.

3 Offline Tracking

Due to space restrictions, we defer to the full version [4] our discussion of the offline problem. We describe a simple greedy approach to solve the problem, either in the simple coverage or c-coverage cases, by choosing at each handover the region that will cover the trajectory the longest. Intuitively, the time the moving point spends within each region may be viewed as forming a set of intervals of the real timeline, and we are applying a standard greedy algorithm to find the smallest subset of the intervals that covers the timeline. We prove the following:

Theorem 1. *The greedy algorithm solves the offline tracking problem optimally, in polynomial time.*

4 Online Tracking

We now move on to the dynamic setting. We assume that we are given the start locations of the trajectory, and receive a sequence of updates extending the trajectory. From these updates we can easily generate a sequence of *events* caused when the trajectory crosses into or out of a region. We will describe three algorithms for different settings, which are all based on the following observations.

Let T be the (unknown) trajectory of our moving entity, and recall that $T(\tau)$ denotes the point in space that the entity occupies at time τ. Let τ_0 be the starting time. We will define a sequence of times τ_i as follows. For any i, let $p_i = T(\tau_i)$ be the location of the entity at time τ_i, and let $\mathscr{D}_i \subset \mathscr{D}$ be the set of regions that contain p_i. For each $D_{ij} \in \mathscr{D}_i$, let τ'_{ij} be first the time after τ_i that the entity leaves D_{ij}. Now, let $\tau_{i+1} = \max_j \tau'_{ij}$ be

the moment that the entity leaves the last of the regions in $\mathscr{D}_i$ (note that it may have re-entered some of the regions). Figure 2 shows an example. Let τ_k be the last assigned time (that is, the entity does not leave all disks $\mathscr{D}_k$ before the the end of its trajectory).

Observation 2 *Any tracking sequence S for trajectory T must have length at least k.*

Proof. For any i, a solution must have a region of $\mathscr{D}_i$ at time τ_i. However, since by construction there is no region that spans the entire time interval $[\tau_i, \tau_{i+1} + \varepsilon]$ (for any $\varepsilon > 0$), there must be at least one handover during this time, resulting in at least $k-1$ handovers, and at least k regions. □

Randomized Tracking with Logarithmic Competitive Ratio. With this terminology in place, we are now ready to describe our randomized algorithm. We begin by computing τ_0, p_0 and $\mathscr{D}_0$ at the start of T. We will keep track of a set of candidate regions $\mathscr{C}$, which we initialize to $\mathscr{C} = \mathscr{D}_0$, and select a random element from the candidate set as the first region to track the entity. Whenever the trajectory leaves its currently assigned region, we compute the subset $\mathscr{C} \subset \mathscr{D}_i$ of all regions that contain the whole trajectory from p_i to the event point, and if $\mathscr{C}$ is not empty we select a new region randomly from $\mathscr{C}$. When $\mathscr{C}$ becomes empty, we have found the next point p_{i+1}, giving us a new nonempty candidate set $\mathscr{C}$. Intuitively, for each point p_i, if the set of candidate regions containing p_i is ordered by their exit times, the selected regions form a random increasing subsequence of this ordering, which has expected length $O(\log \Delta)$, whereas the optimal algorithm incurs a cost of one for each point p_i. Refer to the full version for a more formal description of the algorithm, as well as the proof of the following lemma.

Lemma 1. *The randomized algorithm produces a valid solution of expected length $O(k \log \Delta)$.*

Combining Observation 2 and Lemma 1, we see that the algorithm has a competitive ratio of $O(\log \Delta)$.

Deterministic Tracking with Linear Competitive Ratio. We now describe a deterministic variant of the algorithm. The only thing we change is that, instead of selecting a random member of the set $\mathscr{C}$ of candidate regions, we select an arbitrary element of this set. Here we assume that $\mathscr{C}$ is represented in some deterministic way that we make no further assumptions about. For example, if the elements in $\mathscr{D}$ are unit disks we might store them as a sorted list by the x-coordinate of their center points.

This strategy may seem rather naïve, and indeed produces a competitive ratio that is exponentially larger than that of the randomized strategy of the previous section. But we will see in Section 5 that this is unavoidable, even for the specific case of unit disks.

Again, the full version contains a more formal description of the algorithm, as well as the proof of the following lemma.

Lemma 2. *The deterministic algorithm produces a valid solution of length $O(k\Delta)$.*

As before, combining Observation 2 and Lemma 2, we see that this algorithm has a competitive ratio of $O(\Delta)$.

Deterministic Tracking in One Dimension. In the 1-dimensional case, a better deterministic algorithm is possible. In this case, the regions of $\mathscr{D}$ can only be connected intervals, due to our assumptions that they are closed connected subsets of $\mathbb{R}$.

Now, when we want to pick a new sensor, we have to choose between $c = |\mathscr{C}|$ intervals that all contain the current position of the entity. For each interval C_i, let ℓ_i be the number of intervals in $\mathscr{C} \setminus \{C_i\}$ that contain the left endpoint of C_i, and let r_i be the number of intervals in $\mathscr{C} \setminus \{C_i\}$ that contain the left endpoint of C_i. We say that an interval C_i is *good* if $\max(\ell_i, r_i) \leq c/2$. Our deterministic algorithm simply chooses a good sensor at each step. Figure 3 illustrates this.

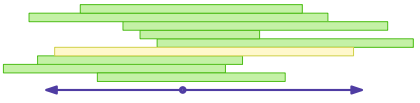


Fig. 3. A set of 8 intervals covering the current location of the entity (blue dot). A good interval is highlighted; this interval has $\ell_i = 3 \leq 8/2$ and $r_i = 2 \leq 8/2$.

The new algorithm is described in the full version, where we also prove the following lemma.

Lemma 3. *The deterministic one-dimensional algorithm produces a valid solution of length* $O(k \log \Delta)$.

Combining Observation 2 and Lemma 3, we conclude that this algorithm also has a competitive ratio of $O(\log \Delta)$.

Summary of Algorithms. Our input assumptions ensure that any trajectory can be transformed in polynomial time into a sequence of events: trivially, for each piece in the piecewise description of the trajectory, we can determine the events involving that piece in time $O(n)$ (where $n = |\mathscr{D}|$) and sort them in time $O(n \log n)$.

Once this sequence is known, it is straightforward to maintain both the set of regions containing the current endpoint of the trajectory, and the set $\mathscr{C}$ of candidate regions, in constant time per event. Additionally, each event may cause our algorithms to select a new region, which may in each case be performed given the set $\mathscr{C}$ in time $O(|\mathscr{C}|) = O(\Delta)$. Therefore, if there are m events in the sequence, the running time of our algorithms (once the event sequence is known) is at most $O(m\Delta)$.

Additionally, geometric data structures (such as those for point location among fat objects [13]) may be of use in more quickly finding the sequence of events, or for more quickly selecting a region from $\mathscr{C}$; we have not carefully analyzed these possibilities, as our focus is primarily on the competitive ratio of our algorithms rather than on their running times.

We summarize these results in the following theorem:

Theorem 3. *Given a set* $\mathscr{D}$ *of* n *connected regions in* $\mathbb{R}^d$*, and a trajectory* T*,*

- *there is a randomized strategy for the online tracking problem that achieves a competitive ratio of* $O(\log \Delta)$*; and*

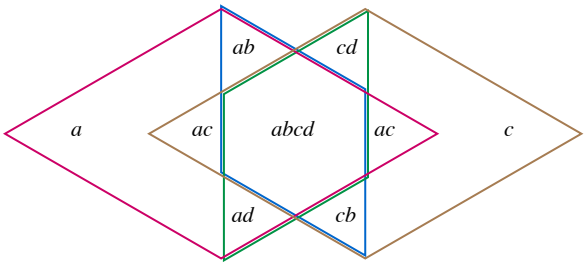


Fig. 4. Four similar rhombi form a set of regions for which no stateless algorithm can be competitive

– *there are deterministic strategies for the online tracking problem that achieve a competitive ratio of* $O(\log\Delta)$ *when* $d = 1$ *or* $O(\Delta)$ *when* $d > 1$.

Each of these strategies may be implemented in polynomial time.

5 Lower Bounds

We now provide several lower bounds on the best competitive ratio that any deterministic or randomized algorithm can hope to achieve. Our lower bounds use only very simple regions in $\mathscr{D}$: similar rhombi, in one case, unit disks in $\mathbb{R}^d$ in a second case, and unit intervals in $\mathbb{R}$ in the third case. These bounds show that our algorithms are optimal, even with strong additional assumptions about the shapes of the regions.

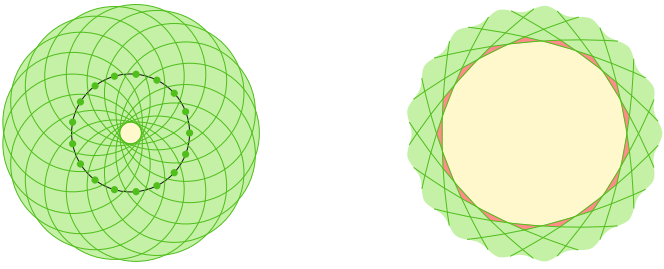
Lower Bounds on Stateless Algorithms. An algorithm is *stateless* if the next sensor that covers the moving point, when it moves out of range of its current sensor, is a function only of its location and not of its previous state or its history of motion. Because they do not need to store and retrieve as much information, stateless algorithms provide a very enticing possibility for the solution of the online tracking problem, but as we show in this section, they cannot provide a competitive solution.

Theorem 4. *There exists a set* $\mathscr{D}$ *of four similar rhombi in* $\mathbb{R}^2$*, such that any stateless algorithm for the online tracking problem has unbounded competitive ratio.*

Proof. The set $\mathscr{D}$ is shown in Figure 4. It consists of four rhombi a, b, c, and d; these rhombi partition the plane into regions (labeled in the figure by the rhombi containing them) such that the common intersection $abcd$ of the rhombi is directly adjacent to regions labeled ab, ac, ad, bc, and cd.

Let G be a graph that has the four rhombi as its vertices, and the five pairs ab, ac, ad, bc, and cd as its edges. Let A be a stateless algorithm for $\mathscr{D}$, and orient the edge xy of G from x to y if it is possible for algorithm A to choose region y when it performs a handover for a trajectory that moves from region $abcd$ to region xy. If different trajectories would cause A to choose either x or y, orient edge xy arbitrarily.

Because G has four vertices and five edges, by the pigeonhole principle there must be some vertex x with two outward-oriented edges xy and xz. There exists a trajectory



(a) (b)

Fig. 5. (a) A set of Δ disks whose centers are equally spaced on a circle. (b) The heart of the construction, zoomed in. The yellow cell is inside all disks; the red cells are inside all but one disk.

T that repeatedly passes from region *abcd* to *xy*, back to *abcd*, to *xz*, and back to *abcd*, such that on each repetition algorithm A performs two handovers, from z to y and back to z. However, the optimal strategy for trajectory T is to cover the entire trajectory with region x, performing no handovers. Therefore, algorithm A has unbounded competitive ratio. □

Lower Bounds on Deterministic Algorithms. Next, we show that any deterministic algorithm in two or more dimensions must have a competitive ratio of Δ or larger, matching our deterministic upper bound and exponentially worse than our randomized upper bound. The lower bound construction consists of a set of Δ unit disks with their centers on a circle, all containing a common point (Figure 5). The idea is that if the trajectory starts at this common point, it can exit from any single disk, in particular, the one that a deterministic algorithm previously chose.

Theorem 5. *There exists a set $\mathscr{D}$ of unit disks in $\mathbb{R}^2$, such that any deterministic algorithm for the online tracking problem has competitive ratio at least $\Delta-1$.*

Proof. Let $\mathscr{D}$ be a set of Δ unit disks whose centers are equally spaced on a given circle C of slightly less than unit radius, as in Figure 5(a). Let the moving point to be tracked start at the point p_0 at the center of C, in the common interior of all disks. For each disk $D_i \in \mathscr{D}$, there exists a cell X_i in the arrangement that is interior to all disks in $\mathscr{D} \setminus \{D_i\}$, but outside D_i itself. Furthermore, this cell is directly adjacent to the center cell. See Figure 5(b) for an illustration.

Now, let A be any deterministic algorithm for the online tracking problem, and construct a sequence of updates to trajectory T as follows. Initially, T consists only of the single point p_0. At each step, let algorithm A update its tracking sequence to cover the current trajectory, let D_i be the final region in the tracking sequence constructed by algorithm A, and then update the trajectory to include a path to X_i and back to the center.

Since X_i is not covered by D_i, algorithm A must increase the cost of its tracking sequence by at least one after every update. That is, $|S_A(T)| \geq |T|$. However, in the optimal tracking sequence, every $\Delta-1$ consecutive updates can be covered by a single region D_i, so $S^*(T) \leq |T|/(\Delta-1)$. Therefore, the competitive ratio of A is at least $\Delta-1$. □

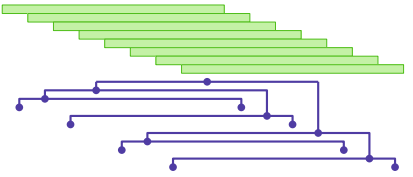


Fig. 6. A set of $\Delta = 8$ intervals, and a tree of 8 different trajectories in $\mathbb{R}^1$ (horizontal dimension)

This construction generalizes to any $d > 2$.

Lower Bounds on Randomized Algorithms. The above lower bound construction uses the fact that the algorithm to solve the problem is deterministic: an adversary constructs a tracking sequence by reacting to each decision made by the algorithm. For a randomized algorithm, this is not allowed. Instead, the adversary must select an entire input sequence, knowing the algorithm but not knowing the random choices to be made by the algorithm. Once this selection is made, we compare the quality of the solution produced by the randomized algorithm to the optimal solution. By Yao's principle [19], finding a randomized lower bound in this model is equivalent to finding a random distribution R on the set of possible update sequences such that, for every possible deterministic algorithm A, the expected value of the competitive ratio of A on a sequence from R is high.

Our lower bound construction consists of Δ unit intervals that contain a common point, and a tree of Δ different possible paths for the moving object to take, each of which leaves the intervals in a different ordering, in a binary tree-like fashion. Half of the trajectories start by going to the left until they are outside the right half of the intervals, the others start towards the right until they are outside the left half of the intervals, and this recurses, as shown in Figure 6.

More formally, let us assume for simplicity that Δ is a power of 2. Let $\mathscr{D}$ be a set of Δ distinct unit intervals in $\mathbb{R}$, containing a common point p_0. For any $k \in [1,\Delta]$ we define point p_k to be a point outside the leftmost k intervals but in the interior of the rest, and p_{-k} to be a point outside the rightmost k intervals but in the interior of the rest.

Now, for each $j \in [1,\Delta]$, we construct a trajectory T_j with $h = \log\Delta$ steps, as follows. We define an index $\xi(j,i)$ for all $j \in [1,\Delta]$ and all $i \in [1,h]$ such that trajectory T_j is at point $p_{\xi(j,i)}$ at step i. At step 0, all trajectories start at $\xi(j,0) = 0$. Then, at step i:

- all T_j with $j \mod 2^{h-i} \leq 2^{h-i-1}$ move to the left to $\xi(j,i) = \min_{l<i} \xi(j,l) - 2^{h-i}$,
- all T_j with $j \mod 2^{h-i} > 2^{h-i-1}$ move to the right to $\xi(j,i) = \max_{l<i} \xi(j,l) + 2^{h-i}$.

Figure 6 shows $\mathscr{T}$ be the resulting set of these Δ trajectories in a tree representation.

Theorem 6. *There exists a set $\mathscr{D}$ of unit intervals in $\mathbb{R}$, for which any randomized algorithm to solve the online tracking problem has competitive ratio $\Omega(\log\Delta)$.*

Proof. Let $\mathscr{D}$ and the set of trajectories $\mathscr{T}$ be as described above. Let R be a probability distribution over the set of all possible trajectories that has a probability of $1/\Delta$ to be any element of $\mathscr{T}$, and a probability of 0 elsewhere.

Now, let A be any deterministic algorithm for the online tracking problem. At each level of the tree, each region D_i that algorithm A might have selected as the final region in its tracking sequence fails to cover one of the two points that the moving point could move to next, and each of these points is selected with probability $1/2$, so algorithm A must extend its tracking sequence with probability $1/2$, and its expected cost on that level is $1/2$. The number of levels is $\log_2 \Delta$, so the total expected cost of algorithm A is $1 + \frac{1}{2}\log_2 \Delta$, whereas the optimal cost on the same trajectory is 1. Therefore the competitive ratio of algorithm A on a random trajectory with distribution R is at least $1 + \frac{1}{2}\log_2 \Delta$.

It follows by Yao's principle that the same value $1 + \frac{1}{2}\log_2 \Delta$ is also a lower bound on the competitive ratio of any randomized online tracking algorithm. □

Although the trajectories formed in this proof are short relative to the size of $\mathscr{D}$, this is not an essential feature of the proof: by concatenating multiple trajectories drawn from the same distribution, we can find a random distribution on arbitrarily long trajectories leading to the same $1 + \frac{1}{2}\log_2 \Delta$ lower bound. This construction generalizes to unit balls in any dimension $d > 1$ as well.

6 Trilateration

We can extend our results to the case where the entity needs to covered with c sensors at any time. We refer to the full version for details; in particular, we prove the following theorems:

Theorem 7. *There exists a randomized algorithm that solves the trilateration problem with a competitive ratio of* $O(\log(\Delta - c))$.

Theorem 8. *There exists a set* $\mathscr{D}$ *of intervals in* $\mathbb{R}$ *of two different lengths, for which any randomized algorithm to solve the online tracking problem has competitive ratio* $\Omega(\log(\Delta - c))$.

7 Conclusions

We studied the online problem of tracking a moving entity among sensors with a minimal number of handovers, combining the kinetic data and online algorithms paradigms. We provided several algorithms with optimal competitive ratios. Interestingly, randomized strategies are able to provably perform significantly better than deterministic strategies, and arbitrarily better than stateless strategies (which form a very natural and attractive class of algortihms in our application).

We are able to track multiple entities using the same algorithms, by simply treating them independently. As a future direction of research, it would be interesting to study the situation where each sensor has a maximum capacity C, and cannot track more than C different entities at the same time. Another possible direction of research is to analyze and optimize the running times of our strategies for particular classes of region shapes or trajectories, something we have made no attempt at.

References

1. Alon, N., Smorodinsky, S.: Conflict-free colorings of shallow discs. In: Proc. 22nd Symp. on Computational Geometry (SoCG), pp. 41–43. ACM, New York (2006)
2. Cho, M., Mount, D., Park, E.: Maintaining Nets and Net Trees under Incremental Motion. In: Dong, Y., Du, D.-Z., Ibarra, O. (eds.) ISAAC 2009. LNCS, vol. 5878, pp. 1134–1143. Springer, Heidelberg (2009)
3. Eppstein, D., Goodrich, M.T.: Studying (non-planar) road networks through an algorithmic lens. In: GIS 2008: Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 1–10 (2008)
4. Eppstein, D., Goodrich, M.T., Löffler, M.: Tracking moving objects with few handovers, arXiv.org/abs/1105.0392 (2011)
5. Eppstein, D., Goodrich, M.T., Trott, L.: Going off-road: transversal complexity in road networks. In: Proc. 17th ACM SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems (ACM GIS), pp. 23–32. ACM, New York (2009)
6. Ghica, O., Trajcevski, G., Zhou, F., Tamassia, R., Scheuermann, P.: Selecting Tracking Principals with Epoch Awareness. In: Proc. 18th ACM SIGSPATIAL Internat. Conf. on Advances in Geographic Information Systems, ACM GIS (2010)
7. Guibas, L., Hershberger, J., Suri, S., Zhang, L.: Kinetic Connectivity for Unit Disks. Discrete Comput. Geom. 25(4), 591–610 (2001)
8. Guibas, L.J.: Kinetic data structures — a state of the art report. In: Agarwal, P.K., Kavraki, L.E., Mason, M. (eds.) Proc. Workshop Alg. Found. Robot., pp. 191–209 (1998)
9. He, G., Hou, J.: Tracking targets with quality in wireless sensor networks. In: 13th IEEE Conf. on Network Protocols (ICNP), pp. 1–12 (2005)
10. Irani, S., Seiden, S.: Randomized algorithms for metrical task systems. Theor. Comput. Sci. 194(1-2), 163–182 (1998)
11. Miller, G.L., Teng, S.-H., Thurston, W., Vavasis, S.A.: Separators for sphere-packings and nearest neighbor graphs. J. ACM 44, 1–29 (1997)
12. Mount, D.M., Netanyahu, N.S., Piatko, C.D., Silverman, R., Wu, A.Y.: A computational framework for incremental motion. In: Proc. 20th Symp. on Computational Geometry (SoCG), pp. 200–209. ACM, New York (2004)
13. Overmars, M., van der Stappen, F.: Range Searching and Point Location among Fat Objects. J. Algorithms 21(3), 629–656 (1996)
14. Pattem, S., Poduri, S., Krishnamachari, B.: Energy-Quality Tradeoffs for Target Tracking in Wireless Sensor Networks. In: Zhao, F., Guibas, L. (eds.) IPSN 2003. LNCS, vol. 2634, pp. 32–46. Springer, Heidelberg (2003)
15. Sleator, D.D., Tarjan, R.E.: Amortized efficiency of list update and paging rules. Commun. ACM 28, 202–208 (1985)
16. Tekinay, S., Jabbari, B.: Handover and channel assignment in mobile cellular networks. IEEE Communications Magazine 29(11), 42–46 (1991)
17. van Leeuwen, E.J.: Better Approximation Schemes for Disk Graphs. In: Arge, L., Freivalds, R. (eds.) SWAT 2006. LNCS, vol. 4059, pp. 316–327. Springer, Heidelberg (2006)
18. Yang, Z., Liu, Y.: Quality of Trilateration: Confidence-Based Iterative Localization. IEEE Trans. on Parallel and Distributed Systems 21(5), 631–640 (2010)
19. Yao, A.: Probabilistic computations: Toward a unified measure of complexity. In: 18th IEEE Symp. on Foundations of Computer Science (FOCS), pp. 222–227 (1977)
20. Yi, K., Zhang, Q.: Multi-dimensional online tracking. In: Proc. of the 20th ACM-SIAM Symp. on Discrete Algorithms (SODA), pp. 1098–1107. SIAM, Philadelphia (2009)
21. Zhao, F., Shin, J., Reich, J.: Information-driven dynamic sensor collaboration. IEEE Signal Processing Magazine 19(2), 61–72 (2002)