Evolving Distributions Under Local Motion

Aditya Acharya
Department of Computer Science, University of Maryland, College Park, MD, USA

David M. Mount
Department of Computer Science and Institute for Advanced Computer Studies,
University of Maryland, College Park, MD, USA

Abstract

Geometric data sets that arise in modern applications are often very large and change dynamically over time. A popular framework for dealing with such data sets is the evolving data framework, where a discrete structure continuously varies over time due to the unseen actions of an evolver, which makes small changes to the data. An algorithm probes the current state through an oracle, and the objective is to maintain a hypothesis of the data set's current state that is close to its actual state at all times. In this paper, we apply this framework to maintaining a set of n point objects in motion in d-dimensional Euclidean space. To model the uncertainty in the object locations, both the ground truth and hypothesis are based on spatial probability distributions, and the distance between them is measured by the Kullback-Leibler divergence (relative entropy). We introduce a simple and intuitive motion model in which, with each time step, the distance that any object can move is a fraction of the distance to its nearest neighbor. We present an algorithm that, in steady state, guarantees a distance of $O(n)$ between the true and hypothesized placements. We also show that for any algorithm in this model, there is an evolver that can generate a distance of $\Omega(n)$, implying that our algorithm is asymptotically optimal.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Evolving data, tracking, imprecise points, local-motion model, online algorithms

Digital Object Identifier 10.4230/LIPIcs.WADS.2025.4

Related Version *Full Version*: `https://arxiv.org/pdf/2409.11779`

Acknowledgements The authors would like to thank Michael Goodrich for suggesting the question of how to model multidimensional evolving point sets.

1 Introduction

Many modern computational applications are characterized by two qualities: data sets are massive and vary over time. A fundamental question is how to maintain some combinatorial structure that is a function of such a data set. The combination of size and dynamics makes maintaining such a structure challenging. Given the large data sizes, single-shot algorithms may be too slow, and common dynamic algorithms, which support explicit requests for insertions and deletions, may not be applicable because structural changes are unseen by the algorithm.

Anagnostopoulos, Kumar, Mahdian, Upfal, and Vandin introduced a model for handling such data sets, called the *evolving data framework* [2]. In this framework, the structure varies repeatedly over time through the unseen actions of an *evolver*, which makes small, stochastic changes to the data set. The algorithm can probe the current state locally using an *oracle*. With the aid of this oracle, the algorithm maintains a *hypothesis* of the current state of the structure that is as close as possible to its actual state at all times. The similarity between the hypothesis and the current state is measured through some *distance function*. The algorithm's objective is to achieve, in the steady state, a small distance between the hypothesis and the actual state. This framework has been applied to a variety of problems [1, 6, 17, 21, 36].

Consider, for example, sorting. The data consists of a set of objects over some total order. The evolver repeatedly selects a random pair of adjacent objects and swaps them. The oracle is given two objects and returns their relative order. The objective of the algorithm is to maintain an order that is as close to the current state as possible, where the distance is measured in terms of the Kendall tau distance, that is, the number of pairwise order inversions [22]. It has been shown that a Kendall tau distance of $O(n)$ is achievable, and this is optimal [5, 6]. It is noteworthy that the optimal algorithm in the evolving framework is based on a simple quadratic-time sequential algorithm, and not $O(n \log n)$-time algorithms, as one might expect. The sorting problem has been generalized to tracking labels on a tree in [1], laying the foundations for a geometric framework for evolving data.

This paper focuses on the question of how to maintain a set of points whose positions evolve continuously in real d-dimensional space, $\mathbb{R}^d$. In motion tracking applications, object movement is recorded through various technologies, including GPS-enabled mobile devices [31], RFID tags [28], and camera-based sensing [35]. Examples include the movement and migration of animals on land and in oceans, traffic and transport, defense and surveillance, and analysis of human behavior (see, e.g., [13, 24]).

Imprecision and uncertainty are unavoidable problems that arise in evolving systems. Due to sensor latency, the exact location of any object cannot be known with certainty. In the best case, any algorithm can maintain only an approximation to the current state. In order to bound the degree of uncertainty, it is necessary to impose restrictions on object motion. In traditional applications of the evolving data framework, the evolver acts randomly. This is not a reasonable assumption in practice, where moving objects are subject to physical laws or may have a sense of agency [15, 32, 34]. Much work has focused on realistic models of motion, but these can be difficult to analyze theoretically. A widely used model assumes that objects have a maximum speed limit, and as time passes, an object can be inferred to lie within a ball whose radius grows linearly over time based on this speed limit [9, 12, 20]. However, in practice, the speed that any object can move depends on the congestion within its local environment.

In multidimensional space, there is no intrinsic total order, and it is less clear what it means to accurately track imprecise moving objects. Given the inherent uncertainty, we model the location of each object in terms of a spatial probability distribution. The distance between the actual state of the system and our hypothesis is naturally defined as the relative entropy (Kullback-Leibler divergence) between these two distributions. The *Kullback-Leibler* (KL) *divergence* is a fundamental measure in statistics that represents the distance between two probability distributions [23]. It has numerous applications in statistical inference, coding theory, machine learning, among others [10]. In our case, it serves two intuitive purposes. First, it measures how different the truth is from our hypothesis. Second, it can be used to quantify the additional information required to encode the actual location of each object, based on our hypothesis [10].

For the evolution of our data, we adopt a locality-sensitive stochastic motion model. We assume that with each time step, the motion of an object is constrained by its immediate environment, which we call the *local-motion model.* In this model, the distance that any object can move in a single time step is a fixed fraction of the distance to its nearest neighbor. The support of this object's probability distribution is a region of size proportional to the nearest-neighbor distance. This model has the advantage that it does not impose arbitrary speed limits on the movement of objects, it is invariant under transformations that preserve relative distances (e.g., translation, rotation, and uniform scaling), and it satisfies the observed phenomenon that objects in dense environments have less personal space [16, 19], and exhibit slower movement than those in sparse environments [4, 29].

To control the communication complexity in determining object locations, we do not require that the oracle returns the exact object positions. Instead, we assume access to an *oracle* that is given a Euclidean ball and the index i of an object. It returns a pair of bits indicating whether the i-th object lies within this ball and (if so) whether there is any other object of the set within this ball. Since every object's location is subject to uncertainty, the same query on the oracle might result in different outcomes at different times, and our algorithm is robust to such variations.

In our framework, the evolver and the algorithm operate asynchronously and in parallel. With each time step, the evolver selects an arbitrary object of the set and moves it in accordance with the local-motion model. (This need not be random and may even be adversarial.) This information is hidden from our algorithm. The algorithm selects an object and invokes the oracle on this object. Based on the oracle's response, the algorithm updates the current hypothesized distribution for this point. Thus, the evolver and algorithm are involved in a pursuit game, with the evolver incrementally changing object distributions (possibly in an adversarial manner) and the algorithm updating its hypothesized distributions. The algorithm's goal is to minimize the relative entropy between these distributions at all times. Our computational model and a formal statement of our results are presented in Section 2.

2 Problem Formulation and Results

Objects

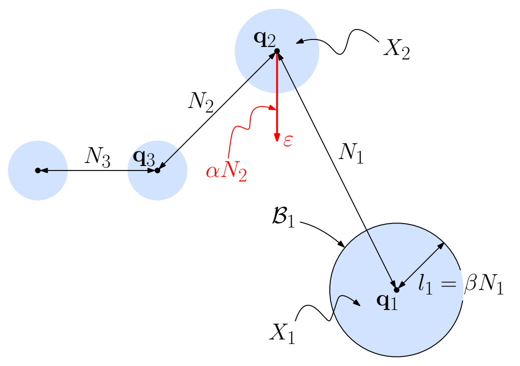
The objects that populate our system can be thought of as imprecise points [8,14,25] that are drawn independently from probability distributions that depend on the proximity to other objects. More formally, at any fixed time step, let $Q = \{\mathbf{q}_1, \ldots, \mathbf{q}_n\} \subset \mathbb{R}^d$ denote a point set of size n in d-dimensional Euclidean space, and let $[n]$ denote the index set $\{1, \ldots, n\}$. For each $\mathbf{q}_i \in Q$, let N_i denote the distance to its nearest neighbor in $Q \setminus \{\mathbf{q}_i\}$. Given $\mathbf{x} \in \mathbb{R}^d$ and nonnegative real r, let $\mathcal{B}(\mathbf{x}, r)$ denote the closed Euclidean ball of radius r centered at $\mathbf{x}$, and let $\mathrm{U}(\mathcal{B}(\mathbf{x}, r))$ denote the uniform probability distribution over this ball. Given a real β, where $0 < \beta < 1$, define the *β-local feature region* of $\mathbf{q}_i$ to be $\mathcal{B}(\mathbf{q}_i, \beta N_i)$. Let $P_i = P_i(\beta)$ denote the uniform[1] probability distribution, $\mathrm{U}\left(\mathcal{B}(\mathbf{q}_i, \beta N_i)\right)$, and let $\mathcal{P} = \{P_1, \ldots, P_n\}$ denote this set of distributions. We refer to $\mathcal{P}$ as the *truth* or *ground truth*, as it represents the true state of the system, subject to the given imprecision. Together, Q and β define a set n independent random variables $\{X_1, \ldots, X_n\}$, with X_i distributed as P_i (see Figure 1a).

The Local-Motion Model

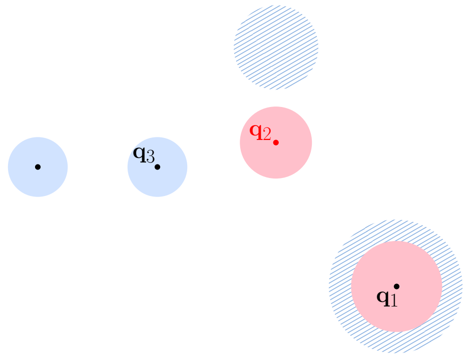
As mentioned in the introduction, there are many ways to model the realistic motion of a collection of objects in an environment. A natural requirement is that each object's motion is affected by the presence of nearby objects. Although there are many ways to incorporate this information (see, e.g., [29]), we have chosen a very simple model, where velocities are influenced by the distance to the nearest neighbor.

We think of the objects of our system as moving continuously over time, and our algorithm queries their state at regular discrete *time steps*. From the algorithm's perspective, objects move, or "evolve," over time in the following manner. Given a parameter α, where $0 < \alpha < 1$, at each time step, an entity called *evolver* selects an object $i \in [n]$ and moves $\mathbf{q}_i$ by a distance of at most αN_i in any direction of its choosing (see Figure 1). While the value of α is known,

[1] Our choice of using a uniform probability distribution is not critical to our approach. In the full version of the paper we show that our results apply to a much broader class of distributions.



(a) Illustration of the objects, and the notations used. The evolver's action ε is shown in red.



(b) Objects that were affected by ε are shown in pink. Tiled regions show the previous state.

Figure 1 The model and an action by the evolver. Shaded regions represent objects.

the action of the evolver, including the choice of the object and the movement, is hidden from the algorithm. Throughout, we assume that the evolver is a *strong adversary*, which means that it has access to our algorithm and the input set [7].

For a nonnegative integer time step t, let $Q^{(t)}$ and $\mathcal{P}^{(t)}$ denote the underlying centers and distributions, respectively, at this time. To simplify notation, we omit the superscript when talking about the current time. We assume that there exists a *bounding region* in the form of a Euclidean ball $\mathcal{B}_0$ centered at the origin. At all times, the points $Q^{(t)}$ are restricted to lie within $\mathcal{B}_0$. The algorithm has knowledge of this ball. Define the system's *initial aspect ratio* to be $\Lambda_0 = (\text{radius}(\mathcal{B}_0))/(\min_{i\in[n]} N_i^{(0)})$. Given any positive constant c, let $c\mathcal{B}_0$ denote a factor-c expansion of this ball about the origin.

Oracle

Consider the state of the system at any fixed time t. Knowledge about the current state of the system is provided by an entity $\mathcal{O}$, called the *oracle*. It is given a Euclidean ball $\mathcal{B}(\mathbf{x}, r)$ and an object index $i \in [n]$. Recall that for each $i \in [n]$, X_i denotes a random variable distributed as P_i. The oracle is a function $\mathcal{O} : [n] \times \mathbb{R}^d \times \mathbb{R}^+ \to \{Y, N\} \times \{+, -\}$, where $\mathcal{O}(i, \mathbf{x}, r)$ returns:

- "Y" or "N" depending on whether $X_i \in \mathcal{B}(\mathbf{x}, r)$ and
- "$+$" or "$-$" depending on whether there exists $j \neq i$, such that $X_j \in \mathcal{B}(\mathbf{x}, r)$

The first element of the pair is used to estimate the location of the object i, and the second is used to estimate the distance to its nearest neighbor. Because X_i and X_j are random variables, so is $\mathcal{O}(i, \mathbf{x}, r)$. Consistent with previous applications of the evolving data framework (see, e.g., [3]), we intentionally made the oracle as weak as possible, implying that our algorithm can be used with stronger oracles.

Hypotheses and Distance

The algorithm maintains a *hypothesis* of the current object locations, which is defined to be a set $\mathcal{H} = \{H_1, \ldots, H_n\}$ of spatial probability distributions in $\mathbb{R}^d$. The *distance* between the truth $\mathcal{P}$ and the current hypothesis $\mathcal{H}$, denoted $\mathcal{D}(\mathcal{P}, \mathcal{H})$, is defined as the sum of n Kullback-Leibler divergences from the hypothesized distributions to the true ones. For $i \in [n]$, let

$$D_i \;=\; D_{\mathrm{KL}}(P_i \parallel H_i) \;=\; \int_{\mathbf{x}\in\mathcal{B}_i} P_i(\mathbf{x}) \log \frac{P_i(\mathbf{x})}{H_i(\mathbf{x})} \mu(d\mathbf{x}),$$

where $\mu(\cdot)$ denotes the measure over $\mathcal{B}_i = \mathcal{B}(\mathbf{q}_i, \beta N_i)$ and define

$$\mathcal{D}(\mathcal{P}, \mathcal{H}) = \sum_{i=1}^{n} D_i = \sum_{i=1}^{n} D_{\mathrm{KL}}(P_i \parallel H_i). \tag{1}$$

As a baseline for comparisons, we introduce a *naïve hypothesis*, denoted $\mathcal{H}^*$, which assumes no information about the locations of the objects, other than the fact that they lie within the bounding region. Recall that $\mathcal{B}_0$ denotes this bounding region and $3\mathcal{B}_0$ denotes a factor-3 expansion about its center. Since all the points of Q lie within $\mathcal{B}_0$ and $0 < \beta < 1$, $3\mathcal{B}_0$ is guaranteed to contain all the β-local feature regions. For all $i \in [n]$, let H_i^* be the uniform distribution over $3\mathcal{B}_0$, and let $\mathcal{H}^* = \{H_1^*, \ldots, H_n^*\}$. Regardless of the initial truth, the initial distance of the ith object satisfies the following bound.

$$\begin{aligned} D_i^* &= \int_{\mathbf{x}\in\mathcal{B}_i} P_i(\mathbf{x}) \log \frac{P_i(\mathbf{x})}{H_i^*(\mathbf{x})} \mu(d\mathbf{x}) = \int_{\mathbf{x}\in\mathcal{B}_i} P_i(\mathbf{x}) \log \frac{1/(\beta N_i)^d}{1/(3 \cdot \mathrm{radius}(\mathcal{B}_0))^d} \mu(d\mathbf{x}) \\ &= \log \left(\frac{3 \cdot \mathrm{radius}(\mathcal{B}_0)}{\beta N_i} \right)^d \int_{\mathbf{x}\in\mathcal{B}_i} P_i(\mathbf{x}) \mu(d\mathbf{x}) \\ &\leq d \left(\log \Lambda_0 + \log \frac{3}{\beta} \right). \end{aligned}$$

Let $\mathcal{D} = \sum_{i\in[n]} D_i^*$. Under our assumption that d and β are constants, we have $\mathcal{D}^* \in \Theta(n \log \Lambda_0)$. The initial aspect ratio, Λ_0, depends on the initial configuration of the points of Q, and can be arbitrarily high. In the best case, when the points are uniformly distributed in $\mathcal{B}_0$, we have $\Lambda_0 = \Omega(n^{1/d})$ and hence $\mathcal{D} \in \Omega(n \log n)$. The objective of our algorithm is to maintain a significantly smaller bound on this distance.

The combination of evolver, oracle, and distance function constitute a model of evolving motion, which we henceforth call the (α, β)*-local-motion model.*

Class of Algorithms

We assume that the algorithm that maintains the hypothesis runs in discrete steps over time. With each step, it may query the oracle a constant number of times, perform a constant amount of work, and then update the current set of hypotheses. The number of oracle queries is independent of n but can depend on dimension d, local feature scale factor β, and motion factor α. In our case, this work takes the form of updating the hypothesis for the object that was queried. In the purest form of the evolving data framework, the evolver and algorithm alternate [2]. Instead, similar to the generalized framework proposed in [1], we assume that there is a fixed *speed-up factor*, denoted σ. When amortized over the entire run, the ratio of the number of steps taken by the algorithm and the evolver does not exceed σ.

Objective and Results

The objective of the algorithm is as follows. Given any starting ground-truth configuration, after an initial "burn-in" period, the algorithm guarantees that the hypothesis is within a bounded distance of the truth subject to model assumptions and the given speed-up factor. Our main result is that there exists an algorithm with constant speed-up factor σ that maintains a distance of $O(n)$ in steady state.

▶ **Theorem 1.** *Consider a set of n evolving objects in $\mathbb{R}^d$ under the (α, β)-local-motion model, for constants α and β, where $0 < \alpha, \beta < \frac{1}{3}$. There exists an algorithm of constant speed-up σ, and burn-in time $t_0 \in O(n \log \Lambda_0 \log\log \Lambda_0)$ such that for all $t \geq t_0$, this algorithm maintains a distance of $O(n)$.*

The algorithm and its analysis will be proved in Section 4. Given that no algorithm can guarantee an exact match between hypothesis and truth, it is natural to wonder how close this is to optimal. In Section 3, we will show that it is asymptotically optimal by showing that for any algorithm and any constant speed-up factor, there exists an evolver that can force a distance of $\Omega(n)$ in steady state.

Why the KL Divergence?

The principal challenge in generalizing the evolving framework from simple 1-dimensional applications like sorting to a multidimensional setting is the lack of an obvious distance measure that captures how close the hypothesis is to the current state. Our approach is motivated by an information-theoretic perspective. The KL divergence $D_i = D_{\mathrm{KL}}(P_i \parallel H_i)$ serves as a measure of how different the actual object distribution P_i is from our hypothesis H_i. (Note that the KL divergence is asymmetric, which is to be expected, given the asymmetric roles of the truth and our hypothesis.)

As an example of this information-theoretic approach, consider the following application in coding theory and space quantization [10]. The objective is to communicate the location of an imprecise point X_i with any arbitrary resolution δ. From Shannon's source coding theorem [26], the theoretical lower bound for the expected number of bits required for communication is given by the *Shannon entropy* [30]

$$b_{P_i} = \sum_{C_\delta \in B_i} P_i(C_\delta) \log \frac{1}{P_i(C_\delta)},$$

where C_δ is a cell (a d-dimensional box) of size δ in $\mathbb{R}^d$, and $P(C)$, by a slight abuse of notation, is the probability associated with a cell C using the underlying probability distribution function P. This is roughly achieved by a Huffman coding [18] of the space around q_i, which intuitively assigns a number of bits proportional to the log of the inverse of the probability measure associated with an event, which in our case is X_i lying within a cell C_δ of size δ.

Since the P_i's are not available to us, we use a similar strategy of encoding the space using H_i. Therefore, in expectation, we use roughly

$$b_{H_i} = \sum_{C_\delta \in B_i} P_i(C_\delta) \log \frac{1}{H_i(C_\delta)}$$

bits. When δ is arbitrarily small, the difference $(b_{H_i} - b_{P_i}) \sim D_i$. Therefore, $\mathcal{D}(\mathcal{P}, \mathcal{H}) = \sum_{i \in [n]} D_i$ roughly represents the expected number of additional bits needed to represent the location of X_i's individually using just H_i's, compared to the absolute theoretical limit.

Paper Organization

The remainder of the paper is organized as follows. In the next section, we present Theorem 2, which provides a lower bound on the distance achievable by any algorithm in our model. In Section 4, we present the algorithm and analyze its steady-state performance. In Section 5, we discuss possible relaxations to the assumptions made in our model.

3 Lower Bound

In this section, we show that maintaining $\mathcal{D}(\mathcal{P}, \mathcal{H}) \in O(n)$ is the best we can hope to achieve. This is established in the following theorem. Due to space limitations, the proof has been deferred to Section A.1.

▶ **Theorem 2.** *For any algorithm $\mathcal{A}$, there exists a starting configuration $Q^{(0)}$ and an evolver (with knowledge of $\mathcal{A}$) in the local-motion model such that, for any positive integer t_0, there exists $t \geq t_0$, such that $\mathcal{D}^{(t)} = \mathcal{D}(\mathcal{P}^{(t)}, \mathcal{H}^{(t)}) \in \Omega(n)$.*

The proof follows a similar structure to other lower-bound proofs in the evolving data framework [1,3,6]. Intuitively, over a period of time of length cn, for $c < 1$, the algorithm can inspect the locations of only a constant fraction of points. During this time, the evolver can move a sufficient number of uninspected points so that the overall distance increases by $\Omega(n)$.

4 Algorithm

In this section, we present our algorithm and analyze its performance. We begin by defining our hypothesis. For every index i, we define two parameters: $h_i \in \mathbb{R}$, and $\mathbf{k}_i = (k_{i,1}, \ldots k_{i,d}) \in \mathbb{R}^d$. For $\mathbf{x} = (x_1, \ldots x_d) \in \mathbb{R}^d$ we set the hypothesis probability density for the ith point to be the d-fold product of independent Cauchy distributions, one per coordinate, with center at $\mathbf{k}_i$ and scaling parameter h_i [33]. That is,

$$H_i(\mathbf{x}) \;=\; f_{h_i,\mathbf{k}_i}(\mathbf{x}) \;=\; \frac{1}{\pi^d h_i^d} \prod_{j=1}^{d} \left(1 + \frac{(x_j - k_{i,j})^2}{h_i^2}\right)^{-1} \qquad \text{(2: Hypothesis Def.)}$$

(Note that this differs from the standard multivariate Cauchy [33].) If we let $f_{1,\mathbf{0}}(\mathbf{x})$ denote the probability density function for the standard d-fold product of Cauchy distributions (centered at the origin unit scale), we can express this equivalently as

$$f_{h_i,\mathbf{k}_i}(\mathbf{x}) \;=\; \frac{1}{h_i^d} f_{1,\mathbf{0}}\left(\frac{\mathbf{x} - \mathbf{k}_i}{h_i}\right), \quad \text{where} \quad f_{1,\mathbf{0}}(\mathbf{x}) \;=\; \frac{1}{\pi^d} \prod_{j=1}^{d} \left(1 + x_j^2\right)^{-1}.$$

It will be convenient to define the *hypothesis ball* $\mathcal{B}_i^H = \mathcal{B}(\mathbf{k}_i, h_i)$, which we use to illustrate H_i. Note that, unlike our truth probabilities P_i, our hypothesis distributions have unbounded support. Our algorithm modifies the parameters $\mathbf{k}_i$ and h_i in response to information received from the oracle.

4.1 Potential Function

Due to the subtleties of tracking the Kullback-Leibler divergence, we introduce a potential function Φ to aid in the analysis. For each object index $i \in [n]$, we define an individual potential function Φ_i, which bounds the distance D_i to within a constant. The total potential Φ will be the sum of these functions.

Let $s_i = \|\mathbf{q}_i - \mathbf{k}_i\|$ denote the Euclidean distance between the center of the actual distribution and the center of the hypothesis ball. Let $l_i = \beta N_i$ denote the radius of the local feature of $\mathbf{q}_i$. Recall that h_i is the radius of the hypothesis ball (see Figure 2). We define the individual and system *potentials* to be

$$\Phi_i \;=\; \Phi(P_i, H_i) \;=\; \log\left(\frac{\max(s_i, l_i, h_i)}{\sqrt{l_i h_i}}\right) \quad \text{and} \quad \Phi \;=\; \Phi(\mathcal{P}, \mathcal{H}) \;=\; \sum_{i \in [n]} \Phi_i \qquad (3)$$

The following lemma shows that, up to additive and scalar constants that depend on the dimension, distances can be bounded above by this potential.

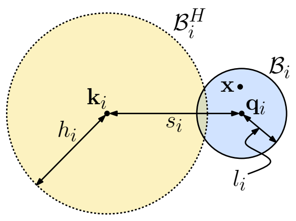


Figure 2 The definition of the individual potential Φ_i.

▶ **Lemma 3.** *There exists a constant c_d (depending on dimension) such that for any $i \in [n]$, $D_i \le c_d(1 + \Phi_i)$, and hence $\mathcal{D} \le c_d(n + \Phi)$.*

Proof. Recall that $P_i(\mathbf{x})$ is the probability density function of a uniform distribution over a ball of radius l_i. Thus, for $\mathbf{x} \in \mathcal{B}_i$, $P_i(\mathbf{x}) = \omega_d / l_i^d$, where ω_d denotes the volume of the unit Euclidean ball in $\mathbb{R}^d$, and $\mathcal{B}_i$. Therefore,

$$\begin{aligned} D_i &= \int_{\mathcal{B}_i} P_i(\mathbf{x}) \log \frac{P_i(\mathbf{x})}{H_i(\mathbf{x})} \mu(d\mathbf{x}) \\ &= \int_{\mathcal{B}_i} P_i(\mathbf{x}) \log \frac{\omega_d / l_i^d}{(\pi^d h_i^d)^{-1} \prod_{j=1}^d \left(1 + \frac{(x_j - k_{i,j})^2}{h_i^2}\right)^{-1}} \mu(d\mathbf{x}) \\ &= \int_{\mathcal{B}_i} P_i(\mathbf{x}) \log\left(\omega_d \pi^d\right) \mu(d\mathbf{x}) \; + \; \int_{\mathcal{B}_i} P_i(\mathbf{x}) \log \frac{h_i^d \prod_{j=1}^d \left(1 + \frac{(x_j - k_{i,j})^2}{h_i^2}\right)}{l_i^d} \mu(d\mathbf{x}). \qquad (4) \end{aligned}$$

For $\mathbf{x} \in \mathcal{B}_i$ and $j \in [d]$, by the triangle inequality, we have

$$|x_j - k_{i,j}| \;\le\; \|\mathbf{x} - \mathbf{k}_i\| \;\le\; \|\mathbf{x} - \mathbf{q}_i\| + \|\mathbf{q}_i - \mathbf{k}_i\| \;\le\; l_i + s_i.$$

Therefore,

$$\begin{aligned} D_i &\le \log\left(\omega_d \pi^d\right) \; + \; \int_{\mathbf{x} \in \mathcal{B}_i} P_i(\mathbf{x}) \log \left[\frac{h_i^d}{l_i^d} \prod_{j=1}^d \left(1 + \frac{(s_i + l_i)^2}{h_i^2}\right) \right] \mu(d\mathbf{x}) \\ &\le \log\left(\omega_d \pi^d\right) \; + \; \log \left[\frac{h_i^d}{l_i^d} \left(1 + \frac{(s_i + l_i)^2}{h_i^2}\right)^d \right] \int_{\mathbf{x} \in \mathcal{B}_i} P_i(\mathbf{x}) \mu(d\mathbf{x}) \\ &= \log\left(\omega_d \; \pi^d\right) \; + \; d \log \left(\frac{h_i^2 + (s_i + l_i)^2}{l_i \; h_i} \right) \;\le\; c_d \left(1 + \log \frac{\max(s_i, l_i, h_i)}{\sqrt{l_i h_i}}\right), \end{aligned}$$

for a suitably chosen c_d that depends only on the dimension. ◀

▶ Remark (Potential function and approximating pairwise distances). The potential function Φ_i we define here is symmetric. It is remarkable that a symmetric function bounds an asymmetric function D_i under the choice of Cauchy distributions as hypotheses.

To demonstrate the value of our potential function, suppose that for $\forall i, \Phi_i < \log c$, for some constant c. This implies that $h_i < c\sqrt{l_i h_i}$, which further implies $h_i < c' l_i$ for some constant c'. Similarly, $s_i < c\sqrt{l_i h_i} < c'' l_i$ for some constant c''. By the triangle inequality, the distance between the centers of two hypothesis balls satisfies

$$\|\mathbf{k}_i - \mathbf{k}_j\| \;\le\; s_i + s_j + \|\mathbf{q}_i - \mathbf{q}_j\| \;\le\; c'' l_i + c'' l_j + \|\mathbf{q}_i - \mathbf{q}_j\|.$$

Since $l_i = N_i / \beta \le \|\mathbf{q}_i - \mathbf{q}_j\| / \beta$, we have $\|\mathbf{k}_i - \mathbf{k}_j\| \le c^* \|\mathbf{q}_i - \mathbf{q}_j\|$, for some constant c^*.

Since $\|\mathbf{k}_i - \mathbf{k}_j\|$ is a constant approximation for $\|\mathbf{q}_i - \mathbf{q}_j\|$, it immediately follows that we can maintain a constant-weight approximation of structures like the Euclidean minimum spanning tree and the Euclidean traveling salesman tour on Q by constructing the same on $\mathbf{k}_i$'s, the centers of the hypotheses H_i's. ⌟

An important feature of our choice of a potential function is that the evolver cannot change its value by more than a constant with each step. Recall that the evolver selects an object index i and moves q_i by a distance of most αN_i in any direction. This results in at most an α fraction change in the values of l_i and s_i, and hence the resulting change in Φ_i is bounded by a constant. However, note that the movement of q_i could potentially affect the local feature sizes of a number of other objects in the system. We can show by a packing argument that there is only a constant number of indices j, whose Φ_j value is affected. Furthermore, these values are only changed by a constant. This is encapsulated in the following lemma. Its proof has been deferred to Section A.1.

▶ **Lemma 4.** *Each step of the evolver increases the potential Φ by at most a constant.*

4.2 The Algorithm

Next, we present our algorithm. The algorithm runs in parallel to the evolver, running faster by a speed-up factor of σ (whose value will be derived in our analysis). Each *step* of the algorithm involves a probe of an object by the oracle, and following that, a possible modification of a hypothesis, H_i. The total running time of the algorithm is proportional to the number of steps.

Let us begin with a high-level explanation. The details are provided in Algorithm TRACK-BYZOOM. Recall that we are given a set of n objects, each represented by X_i, a uniform probability distribution over a ball $\mathcal{B}_i = \mathcal{B}(\mathbf{q}_i, \beta N_i)$, where N_i is the distance to its nearest neighbor and β is the local feature scale. The points $\mathbf{q}_i$ are restricted to lie within a bounding ball $\mathcal{B}_0$ centered at the origin. The algorithm maintains a hypothesis in the form of n Cauchy distributions, each represented by a hypothesis ball $\mathcal{B}_i^H = \mathcal{B}(\mathbf{k}_i, h_i)$, where $\mathbf{k}_i$ is the center of the distribution and h_i is the distribution's scaling parameter.

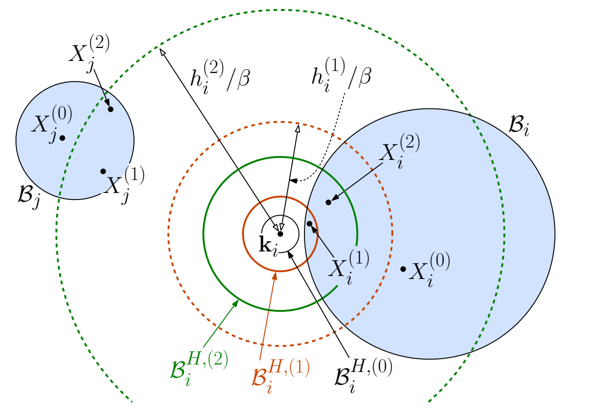
The algorithm begins with an initial hypothesis, where each hypothesis ball is just $\mathcal{B}_0$. This reflects the fact that we effectively assume nothing about the location of $\mathbf{q}_i$, except that it lies within the bounding ball. The algorithm then proceeds in a series of *iterations*, where each iteration handles all the n indices in order. The handling of each index involves two processes, called *zoom-out* and *zoom-in*.

In the zoom-out process, we query the oracle on index i to check whether (1) the sampled point X_i lies within its hypothesis ball and (2) at least one other X_j lies within a concentric ball whose radius is larger by a factor of $\frac{1}{\beta}$. If so, we expand the hypothesis ball by a factor of $3/(1-2\beta)$. (We show in Lemma 5 that this guarantees that the hypothesis ball $\mathcal{B}_i^H$ now contains the local feature ball $\mathcal{B}_i$.) We then proceed to the zoom-in process. If not, we double the hypothesis ball and repeat (see Figure 3).

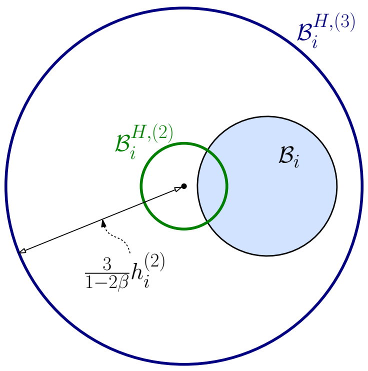
In the zoom-in process, we first check whether X_i lies within the hypothesis ball. (The evolver could have moved q_i.) If not, we return to the zoom-out process. If so, we next check the $\frac{1}{\beta}$-expansion of this ball. If there is no other X_j in this expanded ball, we accept the current hypothesis for i, and go on to the next point in the set. (We show in Lemma 10 that Φ_i at this point in time is bounded by a constant). If there is such an X_j however, we may need to shrink the hypothesis ball for the current index. We cover the hypothesis ball with a collection of $O(1)$ balls whose radii are smaller by a constant factor, and we test whether X_i

Algorithm TrackByZoom TRACKING EVOLVING DISTRIBUTIONS.

```
1: procedure TRACKBYZOOM($\mathcal{O}, n, \beta, \mathcal{B}_0$)
   ▷ Maintain hypothesis $\mathcal{H}$ by running the procedure at a speed-up factor $\sigma$, given the
   membership Oracle:$\mathcal{O}$, the input size:$n$, the local feature parameter:$\beta$, and the the
   maximum bounding ball centered at origin:$\mathcal{B}_0$

2:     for $i \leftarrow 1$ to $n$ do
           ▷ Initially set the hypotheses according to the given bounding ball
3:         $h_i \leftarrow \text{radius}(\mathcal{B}_0)$, $\mathbf{k}_i \leftarrow \mathbf{0}$  ▷ Set $H_i(\mathbf{x}) \leftarrow f_{\mathbf{0},R_0}(\mathbf{x})$, as in Eq.(2: Hypothesis Def.)
4:     end for

5:     $i \leftarrow 1$                                          ▷ Start with the first point

6:     ZOOM_OUT:
7:     $\mathcal{O}_i \leftarrow \mathcal{O}(i, \mathbf{k}_i, h_i)$, $\mathcal{O}_i^* \leftarrow \mathcal{O}(i, \mathbf{k}_i, h_i/\beta)$      ▷ Store values to use later
       ▷ If $\mathcal{B}_i^H$ (hypothesis ball) contains $X_i$, and its $1/\beta$-expansion contains some $X_j$
8:     if $\mathcal{O}_i = (Y, \cdot)$ and $\mathcal{O}_i^* = (Y, +)$ then
9:         $h_i \leftarrow \frac{3}{1-2\beta} h_i$      ▷ Update $\mathcal{B}_i^H$ so that it contains $\mathcal{B}_i$ (local feature of $\mathbf{q}_i$)
10:        go to ZOOM_IN
11:    end if
12:    $h_i \leftarrow 2h_i$                                     ▷ Zoom out by expanding $\mathcal{B}_i^H$
       ▷ Keep zooming out until $X_i \in \mathcal{B}_i^H$, and its $1/\beta$-expansion contains another $X_j$
13:    go to ZOOM_OUT

14:    ZOOM_IN:
15:    $\mathcal{O}_i \leftarrow \mathcal{O}(i, \mathbf{k}_i, h_i)$, $\mathcal{O}_i^* \leftarrow \mathcal{O}(i, \mathbf{k}_i, h_i/\beta)$      ▷ Store values to use later
16:    if $\mathcal{O}_i = (N, \cdot)$ then                      ▷ $X_i$ no longer in $\mathcal{B}_i^H$
17:        go to ZOOM_OUT
18:    end if
       ▷ $X_i \in \mathcal{B}_i^H$, and its $1/\beta$-expansion doesn't contain another $X_j$
19:    if $\mathcal{O}_i = (Y, \cdot)$ and $\mathcal{O}_i^* = (Y, -)$ then
20:        $i \leftarrow 1 + i \mod n$        ▷ $\Phi_i$ is a constant, ready to move to the next point
21:        go to ZOOM_OUT
22:    end if
       ▷ If $\mathcal{B}_i^H$ (hypothesis ball) contains $X_i$, and its $1/\beta$-expansion contains some $X_j$
23:    if $\mathcal{O}_i = (Y, \cdot)$ and $\mathcal{O}_i^* = (Y, +)$ then
           ▷ $\Psi$ is the set of balls of radius $h_i / \left(2\lceil \frac{3}{1-2\beta} \rceil\right)$ which cover $\mathcal{B}_i^H$
24:        for $\mathcal{B}(\mathbf{x}, r) \in \Psi(\mathcal{B}_i^H, 2\lceil \frac{3}{1-2\beta} \rceil)$ do
25:            if $\mathcal{O}(i, \mathbf{x}, r) = (Y, \cdot)$ then      ▷ Found the nested ball that contains $X_i$
26:                $\mathbf{k}_i \leftarrow \mathbf{x}$, $h_i \leftarrow \frac{3}{1-2\beta} r$    ▷ Expand the nested ball so that it contains $B_i$
27:            end if
28:        end for
29:    end if
       ▷ Keep zooming in until $X_i \in \mathcal{B}_i^H$, and its $1/\beta$-expansion contains no other $X_j$
30:    go to ZOOM_IN

31: end procedure
```



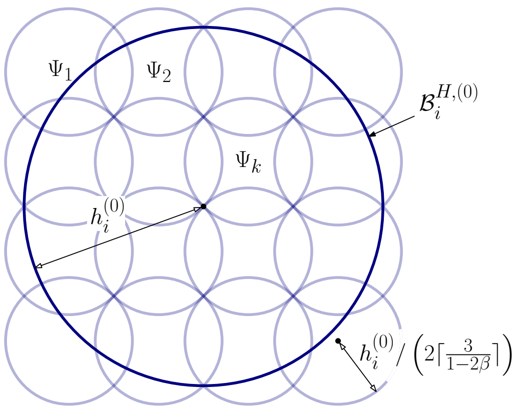
(a) Multiple stages of the zoom-out process starting at $t = 0$. Stage $t = 2$ is the first time $\mathcal{B}_i^{H,(t)}$ contains X_i, and its $1/\beta$-expansion contains X_j.



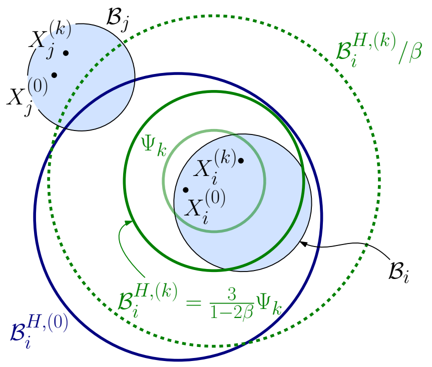
(b) $\mathcal{B}_i^{H,(3)}$ is the hypothesis ball at the end of the zoom-out process. Note that $\mathcal{B}_i \subset \mathcal{B}_i^{H,(3)}$.

Figure 3 The zoom-out process of TRACKBYZOOM (Line 6).

lies within any of them (see Figure 4). As soon as we find one, we shrink the hypothesis ball about this ball. We repeat this process until one of the earlier conditions applies (causing us to return to the zoom-out process or move on to the next point).



(a) Set of nested balls covering $\mathcal{B}_i^{H,(0)}$, which we call Ψ. $\text{radius}(\Psi_i) = \text{radius}(\mathcal{B}_i)/\left(2\lceil\frac{3}{1-2\beta}\rceil\right)$, and $|\Psi| \in O(1)$.



(b) Ψ_k is the first nested ball to contain X_i. $\mathcal{B}_i^{H,(k)}$ contains X_i, and its $1/\beta$-expansion does not contain X_j. TRACKBYZOOM moves on to index $i + 1$.

Figure 4 Illustration of the zoom-in process in TRACKBYZOOM (Line 14).

4.3 Analysis

In this section, we analyze the distance that the algorithm maintains with the ground truth. Recall that the initial hypotheses are based on the bounding ball $\mathcal{B}_0$, which is centered at the origin. Letting $R_0 = \text{radius}(\mathcal{B}_0)$, we have $H_i^{(0)}(\mathbf{x}) = f_{\mathbf{0},R_0}(\mathbf{x})$, as defined in Eq. (2: Hypothesis Def.). Using Eq. (3), the initial potential function satisfies:

$$\Phi_i^{(0)} = \log\sqrt{R_0/l_i^{(0)}}, \quad \text{and} \quad \Phi^{(0)} \in O(n\log\Lambda_0) \qquad (\Lambda_0\text{:Aspect Ratio}) \tag{5}$$

Suppose the condition in line 8, and 23 are satisfied. We prove the following lemma to justify the expansion factor $\frac{3}{1-2\beta}$. The proof is rather technical and has been deferred to the full version of the paper.

▶ **Lemma 5** (Expansion Factor). *If $\mathcal{O}(i, \mathbf{k}_i, h_i) = (Y, \cdot)$ and $\mathcal{O}(i, \mathbf{k}_i, h_i/\beta) = (Y, +)$, then $\mathcal{B}_i \subseteq \mathcal{B}\left(\mathbf{k}_i, \frac{3}{1-2\beta} h_i\right)$.*

Let us consider how Algorithm TrackByZoom affects the potential. For a particular index i, during the *zoom-out* process, we double the radius h_i of the hypothesis ball. If the Euclidean distance between the centers of the hypothesis, and the local-feature ball is much larger than h_i, then $\Phi = \log(\max(s_i, l_i, h_i)/\sqrt{l_i h_i}) = \log(s_i/\sqrt{l_i h_i})$ decreases by an additive constant. However, once h_i is greater than s_i, the algorithm risks increasing Φ. We show that the number of steps where Φ increases is a constant for every index i. However, during the zoom-in process, we maintain the invariant that the hypothesis ball contains the local-feature ball (Lemma 5), which implies that $\max(s_i, l_i, h_i) = h_i$. Therefore, Φ decreases by a constant amount whenever we shrink the hypothesis ball by a constant factor. However, if the evolver moves q_i while index i is being processed, our algorithm may transition to the zoom-out process again. Hence, if the evolver executes some $\varepsilon^{\langle z \rangle}$ steps in the zth iteration, our algorithm may increase Φ by only $O\left(n + \varepsilon^{\langle z \rangle}\right)$. We obtain the following lemma. The proof is deferred to Section A.1

▶ **Lemma 6** (Algorithm's contribution towards Φ). *For any iteration z, let $\varepsilon^{\langle z \rangle}$ denote the number of steps executed by the evolver. Then TrackByZoom increases Φ in $O\left(n + \varepsilon^{\langle z \rangle}\right)$ steps, each time by $O(1)$. In each of the remaining steps of the zth iteration, it decreases Φ by $\Theta(1)$ in an amortized sense.*

Let $\Phi^{\langle z \rangle}$ be the potential function at the start of the zth iteration. And let $\Delta^{\langle z \rangle}$ be the time taken by the zth iteration of the algorithm. At the conclusion of the *zoom-in* process for a particular index i, the algorithm fixes a hypothesis H_i which is at a reasonable distance from the truth P_i. In fact, we show in Lemma 10 that at this point in time, Φ_i is ϕ_0, a constant. Therefore, by Lemma 6, TrackByZoom running at a constant speed-up σ roughly spends $\sim \Phi_i^{\langle z \rangle}$ time (within constant factors), in reducing $\Phi_i^{\langle z \rangle}$ to ϕ_0. Summing over all indices, we see that the time taken by zth iteration, $\Delta^{\langle z \rangle}$, is $\sim \Phi^{\langle z \rangle}$. Intuitively, for a sufficiently large speed-up factor, the actions of the evolver can only affect $\Delta^{\langle z \rangle}$ by a small fraction of this amount. We summarize these observations in the following lemma. The proof is deferred to Section A.1

▶ **Lemma 7** (Iteration time proportional to Potential). *There exists a constant speed-up factor σ for TrackByZoom such that $\Delta^{\langle z \rangle} \in \Theta_\sigma\left(n + \Phi^{\langle z \rangle}\right)$.*

We are now ready to derive how $\Phi^{\langle z \rangle}$ changes over the course of multiple iterations. We show that for a sufficiently large (constant) speed-up factor σ, after $\sim \log\log \Lambda_0$ iterations Φ converges to $O(n)$. (Recall that Λ_0 is the initial aspect ratio.)

▶ **Lemma 8.** *There exists a constant speed-up factor σ for Algorithm TrackByZoom and $z_0 \in \mathbb{Z}$, such that for all $z > z_0$, $\Phi^{\langle z \rangle} \in O(n)$*

Proof. By Lemma 4, in the zth iteration the evolver takes at most $\varepsilon^{\langle z \rangle} = O(\Delta^{\langle z \rangle})$ steps and increases Φ by at most $O(\Delta^{\langle z \rangle})$. Therefore, by Lemma 6, TrackByZoom increases Φ by $O(n + \Delta^{\langle z \rangle})$. Since the algorithm runs at a speed-up factor σ it takes $\sigma\Delta^{\langle z \rangle}$ steps, and by Lemma 6, it decreases Φ by at least $\lambda(\sigma\ \Delta^{\langle z \rangle} - O(n + \Delta^{\langle z \rangle}))$, for some $\lambda \in O(1)$. Accounting for all the increases and decreases we have:

$$\begin{aligned}\Phi^{\langle z+1\rangle} &\leq \Phi^{\langle z\rangle} + O(\Delta^{\langle z\rangle}) + O(n+\Delta^{\langle z\rangle}) - \lambda(\sigma\,\Delta^{\langle z\rangle} - O(n+\Delta^{\langle z\rangle}))\\ &\leq \Phi^{\langle z\rangle} + O(\Delta^{\langle z\rangle}) - \sigma\lambda\Delta^{\langle z\rangle} + O(n).\end{aligned}$$

There exists a sufficiently large $\sigma \in O(1)$ such that $\sigma\lambda\Delta^{\langle z\rangle} - O(\Delta^{\langle z\rangle}) \geq a\Delta^{\langle z\rangle}$, for $0 < a \in O(1)$. So,

$$\Phi^{\langle z+1\rangle} \leq \Phi^{\langle z\rangle} - a\Delta^{\langle z\rangle} + O(n).$$

By Lemma 7, for a sufficiently large (constant) σ, there exists $c_\sigma > 0$ such that $\Delta^{\langle z\rangle} \geq c_\sigma\left(n + \Phi^{\langle z\rangle}\right)$. Therefore,

$$\Phi^{\langle z+1\rangle} \leq (1-ac_\sigma)\Phi^{\langle z\rangle} + (1-ac_\sigma)O(n) \leq (1-ac_\sigma)^{z+1}\Phi^{\langle 0\rangle} + \left(\sum_{y=1}^{z+1}(1-ac_\sigma)^y\right)O(n).$$

By Eq. (5), $\Phi^{\langle 0\rangle} = \Phi^{(0)} \in O(n\log\Lambda_0)$, and since there exists $b > 1$ such that $(1-ac_\sigma) < 1/b < 1$, we have

$$\Phi^{\langle z+1\rangle} \leq \frac{O(n\log\Lambda_0)}{b^{z+1}} + O(n). \qquad (6)$$

Observe that this is $O(n)$ whenever $z > \log\log\Lambda_0$. ◀

If $\Phi^{\langle z\rangle} \in O(n)$, then by Lemma 7, $\Delta^{\langle z\rangle} \in \Theta(n)$. Thus, Φ increases by at most $O(n)$ throughout iteration z. Therefore, for any time t during iteration z, $\Phi^{(t)} \in O(n)$ as well.

Using Eq. (6) we observe that for all $z > 1$, $\Phi^{\langle z\rangle} \leq O(n\log\Lambda_0)$. By Lemma 7, this implies that $\Delta^{\langle z\rangle} \in \Theta(n\log\Lambda_0)$. Hence, for some $t_0 \in O(n\log\Lambda_0 \cdot \log\log\Lambda_0)$, we have $\Phi^{(t)} \in O(n)$, for all $t > t_0$.

By combining Lemmas 8 and 3, we complete the proof of Theorem 1. It is noteworthy that there is a trade-off between the speed-up factor σ and the motion parameter α. Recall that with each step, the evolver can move q_i by a distance of up to αN_i. If we reduce this parameter to $\alpha' = (1+\alpha)^{\frac{1}{c}} - 1$, then it takes the evolver at least c steps to effect the same change as before. Thus, by reducing the local-motion factor, it is possible to achieve a speed-up factor of $\sigma = 1$. Formally we have,

▶ **Corollary 9.** *There exists a motion parameter $\alpha' < 1$ such that for a set of n evolving objects in $\mathbb{R}^d$ under the (α',β)-local-motion model, where $\beta < \frac{1}{3}$, Algorithm* TrackByZoom *(with speed-up factor 1) maintains a distance of $O(n)$ for all $t \geq t_0$, $t_0 \in O(n\log\Lambda_0\log\log\Lambda_0)$*

5 Extensions and Conclusion

The local-motion model introduced in Section 2 assumes a uniform probability distribution for each object. However, the algorithm TrackByZoom with an appropriate speed-up factor can be applied to a wider class of probability distributions. A distribution P_i in this class has $\mathcal{B}_i$, the volume of the ith object, as its support, and is scaled according to l_i, the local feature size. Many natural truncated distributions [11], such as the uniform distribution, the truncated normal distribution [27], the truncated Cauchy distribution belong to this family. Details are presented in the full version of the paper.

An additional refinement to the motion model we suggested might involve redefining the local feature region and the speed of a point based on the distance to its k-nearest neighbors for a fixed k, instead of just using the nearest neighbor distance. We conjecture that our principal results extend to this generalization as well, given a suitable range counting oracle. Yet another variation of the model could possibly involve letting the uncertainty associated with a point have unbounded support (for e.g. a normal rather than a truncated normal distribution). This is a significantly harder problem, as a point may lie outside its local feature with a constant probability, requiring a maintaining algorithm to have a super-constant speed-up. We aim to study these in the future.

References

1 A. Acharya and D. M. Mount. Optimally tracking labels on an evolving tree. In *Proc. 34th Canad. Conf. Comput. Geom.*, pages 1–8, 2022.

2 A. Anagnostopoulos, R. Kumar, M. Mahdian, and E. Upfal. Sorting and selection on dynamic data. *Theoretical Computer Science*, 412(24):2564–2576, 2011. doi:10.1016/j.tcs.2010.10.003.

3 A. Anagnostopoulos, R. Kumar, M. Mahdian, E. Upfal, and F. Vandin. Algorithms on evolving graphs. In *Proc. 3rd Innov. Theor. Comput. Sci. Conf.*, pages 149–160, 2012. doi:10.1145/2090236.2090249.

4 M. Beermann and A. Sieben. The connection between stress, density, and speed in crowds. *Scientific Reports*, 13:13626, 2023. doi:10.1038/s41598-023-39006-8.

5 J. J. Besa, W. E. Devanny, D. Eppstein, M. T. Goodrich, and T. Johnson. Optimally sorting evolving data. In *45th Internat. Colloq. Autom., Lang., and Prog.*, pages 81:1–81:13, 2018. doi:10.4230/LIPIcs.ICALP.2018.81.

6 J. J. Besa, W. E. Devanny, D. Eppstein, M. T. Goodrich, and T. Johnson. Quadratic time algorithms appear to be optimal for sorting evolving data. In *Proc. 20th Workshop Algorithm Eng. and Exp.*, pages 87–96, 2018.

7 A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis.* Cambridge University Press, 2005.

8 K. Buchin, M. Löffler, P. Morin, and W. Mulzer. Delaunay triangulation of imprecise points simplified and extended. *Algorithmica*, 61:674–693, 2011. doi:10.1007/s00453-010-9430-0.

9 D. Busto, W. Evans, and D. Kirkpatrick. Minimizing interference potential among moving entities. In *Proc. 30th Annu. ACM-SIAM Sympos. Discrete Algorithms*, pages 2400–2418, 2019. doi:10.5555/3310435.3310582.

10 T. M. Cover and J. A. Thomas. *Elements of Information Theory.* Wiley, 2012. doi:10.1002/047174882X.

11 Y. Dodge. *The Oxford Dictionary of Statistical Terms.* Oxford Univ. Press, 2003. doi:10.1002/sim.1812.

12 W. Evans and D. Kirkpatrick. Minimizing query frequency to bound congestion potential for moving entities at a fixed target time. *Algorithms*, 17:246, 2024. doi:10.3390/a17060246.

13 J. Gudmundsson, P. Laube, and T. Wolle. Computational movement analysis. In W. Kresse and D. Danko, editors, *Handbook of Geogr. Inf.*, pages 725–741. Springer, 2012. doi:10.1007/978-3-540-72680-7_22.

14 L. J. Guibas, D. Salesin, and J. Stolfi. Epsilon geometry: Building robust algorithms from imprecise computations. In *Proc. Fifth Annu. Sympos. Comput. Geom.*, pages 208–217, 1989. doi:10.1145/73833.73857.

15 D. Helbing, L. Buzna, A. Johansson, and T. Werner. Self-organized pedestrian crowd dynamics: Experiments, simulations, and design solutions. *Transportation Science*, 39:1–24, 2005. doi:10.1287/trsc.1040.0108.

16 O. Hesham and G. Wainer. Context-sensitive personal space for dense crowd simulation. In *Proc. Symp. Simulation for Architecture and Urban Design*, pages 1–8, 2017. URL: https://dl.acm.org/doi/10.5555/3289787.3289806.

17 Q. Huang, X. Liu, X. Sun, and J. Zhang. Partial sorting problem on evolving data. *Algorithmica*, 79:960–983, 2017. doi:10.1007/s00453-017-0295-3.

18 D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40:1098–1101, 1952. `doi:10.1109/JRPROC.1952.273898`.

19 U. Jain. Effects of population density and resources on the feeling of crowding and personal space. *J. Social Psych.*, 127:331–338, 1987. `doi:10.1080/00224545.1987.9713699`.

20 S. Kahan. A model for data in motion. In *Proc. 23rd Annu. ACM Sympos. Theory Comput.*, pages 265–277, 1991. `doi:10.1145/103418.103449`.

21 V. Kanade, N. Leonardos, and F. Magniez. Stable matching with evolving preferences. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, 2016.

22 M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30:81–93, 1938. `doi:10.2307/2332226`.

23 S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22:79–86, 1951. `doi:10.1214/aoms/1177729694`.

24 P. Laube. *Computational Movement Analysis*. Springer Cham, 2014. `doi:10.1007/978-3-319-10268-9`.

25 M. Löffler and M. van Kreveld. Largest and smallest convex hulls for imprecise points. *Algorithmica*, 56:235–269, 2010. `doi:10.1007/s00453-008-9174-2`.

26 D. J. C. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003. URL: `https://books.google.com/books?id=AKuMj4PN_EMC`.

27 F. Nielsen. Statistical divergences between densities of truncated exponential families with nested supports: Duo Bregman and duo Jensen divergences. *Entropy*, 24:421, 2022. `doi:10.3390/e24030421`.

28 R. Pan, Z. Han, T. Liu, H. Wang, J. Huang, and W. Wang. An RFID tag movement trajectory tracking method based on multiple RF characteristics for electronic vehicle identification ITS applications. *Sensors*, 23:7001, 2023. `doi:10.3390/s23157001`.

29 A. Seyfried, B. Steffen, W. Klingsch, and M. Boltes. The fundamental diagram of pedestrian movement revisited. *J. Stat. Mech. Theory Exp.*, 2005:P10002, 2005. `doi:10.1088/1742-5468/2005/10/P10002`.

30 C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:623–656, 1948. `doi:10.1002/j.1538-7305.1948.tb00917.x`.

31 S. M. Tomkiewicz, M. R. Fuller, J. G. Kie, and K. K. Bates. Global positioning system and associated technologies in animal behaviour and ecological research. *Phil. Trans. R. Soc. B*, 365:2163–2176, 2010. `doi:10.1098/rstb.2010.0090`.

32 M. van Kreveld, M. Löffler, F. Staals, and L. Wiratma. A refined definition for groups of moving entities and its computation. *Internat. J. Comput. Geom. Appl.*, 28(2):181–196, 2018. `doi:10.4230/LIPIcs.ISAAC.2016.130`.

33 S. Verdú. The Cauchy distribution in information theory. *Entropy*, 25:346, 2023. `doi:10.3390/e25020346`.

34 L. Wiratma, M. Löffler, and F. Staals. An experimental comparison of two definitions for groups of moving entities. In *Proc. 10th Internat. Conf. Geogr. Inf. Sci.*, pages 64:1–64:6, 2018. `doi:10.4230/LIPIcs.GIScience.2018.64`.

35 S. Yi, H. Li, and X. Wang. Understanding pedestrian behaviors from stationary crowd groups. In *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, pages 3488–3496, 2015. `doi:10.1109/CVPR.2015.7298971`.

36 Y. Zou, G. Zeng, Y. Wang, X. Liu, X. Sun, J. Zhang, and Q. Li. Shortest paths on evolving graphs. In H. T. Nguyen and V. Snasel, editors, *Computational Social Networks*, pages 1–13. Springer, 2016. `doi:10.1007/978-3-319-42345-6_1`.

A Appendix

A.1 Deferred Proofs

Note that some of the restatements of the lemmas given here provide additional details, which were not present in the original statements.

The next lemma applies whenever Algorithm TrackByZoom has completed its processing of an object on Line 20. It shows that the potential value for this object does not exceed a constant.

▶ **Lemma 10** (Done tracking X_i). *There exists a constant ϕ_0, such that whenever Algorithm* TrackByZoom *completes its processing of any object i, $\Phi_i \leq \phi_0 \in O(1)$.*

Proof. The algorithm moves on to the next point when the condition in line 19 is satisfied ($\mathcal{O}(i, \mathbf{k}_i, h_i) = (Y, \cdot)$ and $\mathcal{O}(i, \mathbf{k}_i, h_i/\beta) = (Y, -)$). Since $\mathcal{O}(i, \mathbf{k}_i, h_i) = (Y, \cdot)$, $\mathcal{B}_i^H = \mathcal{B}(\mathbf{k}_i, h_i)$ intersect. Moreover, if $\mathbf{q}_j$ is the current nearest neighbor of $\mathbf{q}_i$, $\mathcal{O}(i, \mathbf{k}_i, h_i/\beta) = (Y, -)$ implies X_j lies outside the $1/\beta$ expansion of $\mathcal{B}_i^H$ (see Figure 5).

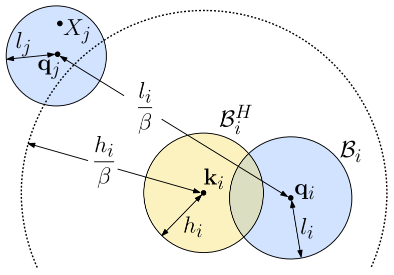


Figure 5 Proof of Lemma 10.

Now because $\mathbf{q}_j$ is the nearest neighbor of $\mathbf{q}_i$, $N_j \leq N_i$, hence $l_j \leq l_i$. By triangle inequality, the following series of inequality holds:

$$\frac{h_i}{\beta} \leq \|X_j - \mathbf{k}_i\| \leq \|X_j - \mathbf{q}_i\| + \|\mathbf{q}_i - \mathbf{k}_i\| \leq \left(\frac{l_i}{\beta} + l_j\right) + (h_i + l_i) \leq \left(\frac{1}{\beta} + 2\right) l_i + h_i,$$

implying that $\frac{1-\beta}{1+2\beta} h_i \leq l_i$.

Now consider the hypothesis ball, $\mathcal{B}_i^{H^-}$, set by the algorithm in the previous step (via Line 26). From Lemma 5, we have $\mathcal{B}_i \subseteq \mathcal{B}_i^{H^-}$. Let $h_i^- = \text{radius}\left(\mathcal{B}_i^{H^-}\right)$. Since the algorithm shrinks the hypothesis ball by at least a factor of 2 during the zoom-in process, h_i^- is at least $2h_i$. And therefore $2h_i \geq l_i$. We also have $s_i \leq h_i + l_i$ (see Figure 5). Therefore,

$$\begin{aligned}\Phi_i = \log\left(\frac{\max(s_i, l_i, h_i)}{\sqrt{l_i h_i}}\right) &\leq \log\left(\frac{l_i + h_i}{\sqrt{l_i h_i}}\right) \\ &\leq \log\left(\frac{2h_i + h_i}{\sqrt{\frac{1-\beta}{1+2\beta} h_i \cdot h_i}}\right) \leq \phi_0, \quad \text{for } \phi_0 = \log\left(3 \cdot \sqrt{\frac{1+2\beta}{1-\beta}}\right).\end{aligned}$$ ◀

▶ **Theorem 2** (Lower Bound on the Distance). *For any algorithm $\mathcal{A}$, there exists a starting configuration $Q^{(0)}$ and an evolver (with knowledge of $\mathcal{A}$) in the local-motion model such that, for any positive integer t_0, there exists $t \geq t_0$, such that $\mathcal{D}^{(t)} = \mathcal{D}(\mathcal{P}^{(t)}, \mathcal{H}^{(t)}) \in \Omega(n)$.*

Proof. We construct a point set on the real line ($\mathbb{R}$), and we will mention at the end how to adapt the proof to any dimension d. Given the small constant β, the local feature constant, we define $Q^{(0)}$ to be the following set of $n = 2m$ points in $\mathbb{R}$.

$$Q^{(0)} \;=\; \bigcup_{i\in[m]} \left\{a_i^{(0)} = 100i\right\} \cup \bigcup_{i\in[m]} \left\{b_i^{(0)} = 100i+1\right\}.$$

Let $\mathcal{D}_{a,i} = \mathcal{D}_{2(k-1)+1}$, $k \in [m]$, the distance corresponding to the first point in the tuple. We similarly define $P_{a,i}$ and $H_{a,i}$. Let N_i denote the current distance to a_i's nearest neighbor. The local feature interval for a_i, denoted I_i, is $100i + \beta N_i[-1,1]$. Recall that $P_{a,i}$ is the uniform probability over I_i. Therefore, $P_{a,i}(\mathbf{x}) = 1/|\mathcal{I}_i|$, where $|\mathcal{I}_i| = 1/\beta N_i$ is the diameter of $\mathcal{I}_i$. Now

$$\begin{aligned}
\mathcal{D}_{a,i}^{(t_0)} \;&=\; \int_{\mathbf{x}\in\mathcal{I}_i} P_{a,i}^{(t_0)}(\mathbf{x}) \log \frac{P_{a,i}^{(t_0)}(\mathbf{x})}{H_{a,i}^{(t_0)}(\mathbf{x})} d\mathbf{x} \\
&=\; \int_{\mathbf{x}\in\mathcal{I}_i} P_{a,i}^{(t_0)}(\mathbf{x}) \log P_{a,i}^{(t_0)}(\mathbf{x}) d\mathbf{x} \;-\; \int_{\mathbf{x}\in\mathcal{I}_i} P_{a,i}^{(t_0)}(\mathbf{x}) \log H_{a,i}^{(t_0)}(\mathbf{x}) d\mathbf{x}.
\end{aligned}$$

By the concavity of the log function and Jensen's inequality, we have

$$\begin{aligned}
\mathcal{D}_{a,i}^{(t_0)} \;&\geq\; -\log|\mathcal{I}_i| \int_{\mathbf{x}\in\mathcal{I}_i} P_{a,i}^{(t_0)}(\mathbf{x}) d\mathbf{x} \;-\; \log \int_{\mathbf{x}\in\mathcal{I}_i} H_{a,i}^{(t_0)}(\mathbf{x}) P_{a,i}^{(t_0)}(\mathbf{x}) d\mathbf{x} \\
&=\; -\log|\mathcal{I}_i| - \log \frac{\int_{\mathbf{x}\in\mathcal{I}_i} H_{a,i}^{(t_0)}(\mathbf{x}) d\mathbf{x}}{|\mathcal{I}_i|} \;=\; -\log \int_{\mathbf{x}\in\mathcal{I}_i} H_{a,i}^{(t_0)}(\mathbf{x}) d\mathbf{x}. \qquad (7)
\end{aligned}$$

We may assume that $\mathcal{D}^{(t_0)} \in o(n)$ (for otherwise we can set $t = t_0$ and are done). This implies there are only $o(n)$ many a_i's such that $\mathcal{D}_{a,i}^{(t_0)} \in \Omega(1)$. Call the remaining $n/2 - o(n)$, a_i's the proximal set, denoted T_a.

We are now ready to describe the evolver's actions. It chooses to stay dormant until time t_0. For a large enough constant M, consider the time interval $[t_0, t_0 + n/M]$. The algorithm $\mathcal{A}$, running at a speed-up factor of σ, can modify at most $\sigma n/M$ hypotheses in the time interval. The evolver chooses not to move any of those points. Therefore $\mathcal{A}$ can only reduce $\mathcal{D}^{(t_0)}$ by $o(n)$ over this time interval. Now there are at least $n/2 - \sigma n/M - o(n)$ members in the proximal set T_a, whose hypotheses were not altered by $\mathcal{A}$. Call this set the stable set, denoted S_a.

For $\kappa = \lceil \log(2/(1+\alpha)) \rceil$, the evolver selects some $n/(\kappa M)$ members from S_a. Call that set S'_a. To be specific, for all $a_i \in S'_a$, the evolver chooses to move a_i away from b_i some κ times, by a distance of exactly $\alpha N_{a,i}$, where $N_{a,i}$ is the distance of a_i from its current nearest neighbor. (Note that the nearest neighbor will remain b_i throughout these operations.) Given this value of κ, after the conclusion of these operations, the distance between a_i and b_i is at least 2. For $\beta < 1/3$, the local feature of a_i changes to an interval $\mathcal{I}'_i$, where $\mathcal{I}'_i \cap \mathcal{I}_i = \emptyset$. Now, for $t = t_0 + n/M$, using a similar analysis as Eq. (7), we have

$$\mathcal{D}_{a,i}^{(t)} \;\geq\; -\log \int_{\mathbf{x}\in\mathcal{I}'_i} H_{a,i}^{(t)}(\mathbf{x}) d\mathbf{x}. \qquad (8)$$

Thus, for all $a_i \in S'_a$, we have $H_{a,i}^{(t)} = H_{a,i}^{(t_0)}$ and $D_{a,i}^{(t_0)} = o(1)$. Therefore,

$$\int_{\mathbf{x}\in\mathcal{I}_i} H_{a,i}^{(t_0)}(\mathbf{x}) d\mathbf{x} \;\geq\; e^{-o(1)}.$$

If $\mathcal{D}_{a,i}^{(t)} \in o(1)$ as well, then similarly from Eq. (8), we have

$$\int_{\mathbf{x}\in\mathcal{I}_i'} H_{a,i}^{(t)}(\mathbf{x})d\mathbf{x} \;\geq\; e^{-o(1)}.$$

But, this yields a contradiction since $\mathcal{I}_i' \cap \mathcal{I}_i = \emptyset$ and $H_{a,i}^{(t)} = H_{a,i}^{(t_0)}$ implies that

$$\int_{\mathbf{x}\in\mathcal{I}_i'\cup\mathcal{I}_i} H_{a,i}^{(t)}(\mathbf{x})d\mathbf{x} \;\geq\; 2e^{-o(1)} \;>\; 1.$$

Therefore, for $a_i \in S_a'$, $\mathcal{D}_{a,i}^{(t)} \in \Omega(1)$, and since $|S_a'| = \Omega(n)$, we have $\mathcal{D}^{(t)} \in \Omega(n)$.

For a general dimension d, we define Q, in the same way except all the points lie on a single axis. The evolver also moves these points along that particular axis. The rest of the analysis involves integration over a region of space rather an interval, but is straightforward and carries over to $\mathbb{R}^d$. ◀

▶ **Lemma 4** (Evolver's contribution to Φ). *Every step of the evolver increases the potential function Φ by at most a constant.*

Proof. Let's say the evolver chooses to move $\mathbf{q}_i$ at time 0. Now $l_i^{(0)}/\beta$ is the nearest neighbor distance of $\mathbf{q}_i$ at time 0. Therefore $\mathbf{q}_i$ moves by at most $(\alpha/\beta)l_i^{(0)}$ distance, implying $s_i^{(1)} \leq s_i^{(0)} + (\alpha/\beta)l_i^{(0)} \leq (1+\alpha/\beta)\max\left(s_i^{(0)}, l_i^{(0)}, h_i^{(0)}\right)$. Similarly $(1-\alpha)l_i^{(0)} \leq l_i^{(1)} \leq (1+\alpha)l_i^{(0)}$. Therefore,

$$\begin{aligned}
\Phi_i^{(1)} - \Phi_i^{(0)} \;&\leq\; \log\frac{\max\left(s_i^{(1)}, l_i^{(1)}, h_i^{(1)}\right)}{\sqrt{l_i^{(1)}h_i^{(1)}}} - \log\frac{\max\left(s_i^{(0)}, l_i^{(0)}, h_i^{(0)}\right)}{\sqrt{l_i^{(0)}h_i^{(0)}}} \\
&\leq\; \log\frac{\max\left(s_i^{(0)} + (\alpha/\beta)l_i^{(0)}, (1+\alpha)l_i^{(0)}, h_i^{(0)}\right)}{\sqrt{(1-\alpha)l_i^{(0)}h_i^{(0)}}} \;-\; \log\frac{\max\left(s_i^{(0)}, l_i^{(0)}, h_i^{(0)}\right)}{\sqrt{l_i^{(0)}h_i^{(0)}}} \\
&\leq\; \log\left(\frac{1+\alpha/\beta}{\sqrt{1-\alpha}}\frac{\max\left(s_i^{(0)}, l_i^{(0)}, h_i^{(0)}\right)}{\sqrt{l_i^{(0)}h_i^{(0)}}}\right) \;-\; \log\frac{\max\left(s_i^{(0)}, l_i^{(0)}, h_i^{(0)}\right)}{\sqrt{l_i^{(0)}h_i^{(0)}}} \\
&=\; \log\frac{1+\alpha/\beta}{\sqrt{1-\alpha}} \in O(1) \qquad (9)
\end{aligned}$$

Let $\mathcal{N}_i$ be the set of indices of points whose nearest neighbor at time 0 was $\mathbf{q}_i^{(0)}$, or whose nearest neighbor at time 1 was $\mathbf{q}_i^{(1)}$. Only points with indices in this set have their potential function changed. If $j \in \mathcal{N}_i$, $s_j^{(0)}$, and $h_j^{(0)}$ do not change. Since $\mathbf{q}_i$ moves by at most $(\alpha/\beta)l_i^{(0)}$ distance, the nearest neighbor distance $N_j = l_j/\beta$ changes by at most $(\alpha/\beta)l_i^{(0)}$ as well. Therefore,

$$-(\alpha/\beta)l_i^{(0)} \;\leq\; N_j^{(1)} - N_j^{(0)} \;\leq\; (\alpha/\beta)l_i^{(0)} \;\implies\; -\alpha l_i^{(0)} \;\leq\; l_j^{(1)} - l_j^{(0)} \;\leq\; \alpha l_i^{(0)} \qquad (10)$$

Since $(1-\alpha)l_i^{(0)} \;\leq\; l_i^{(1)}$, we also have

$$-\frac{\alpha}{1-\alpha}l_i^{(1)} \;\leq\; l_j^{(1)} - l_j^{(0)} \;\leq\; \frac{\alpha}{1-\alpha}l_i^{(1)} \qquad (11)$$

If $\mathbf{q}_i^{(0)}$ was the nearest neighbor of $\mathbf{q}_j^{(0)}$, then $N_j^{(0)} \leq N_i^{(0)}$, and hence $l_j^{(0)} \leq l_i^{(0)}$. Using the last fact, and dividing Eq. (10) by $l_j^{(0)}$, we have $(1-\alpha) \leq l_j^{(1)}/l_j^{(0)} \leq (1+\alpha)$. Similarly, If $\mathbf{q}_i^{(1)}$ was the nearest neighbor of $\mathbf{q}_j^{(1)}$, then using Eq. (11) we have $(1-\alpha/(1-\alpha)) \leq l_j^{(1)}/l_j^{(0)} \leq (1+\alpha/(1-\alpha))$. Therefore $l_j^{(1)}/l_j^{(0)} \in \Theta(1)$ for $j \in \mathcal{N}_i$. And hence,

$$\begin{aligned}
\Phi_j^{(1)} - \Phi_j^{(0)} &= \log \frac{\max\left(s_i^{(1)}, l_i^{(1)}, h_i^{(1)}\right)}{\sqrt{l_i^{(1)} h_i^{(1)}}} - \log \frac{\max\left(s_i^{(0)}, l_i^{(0)}, h_i^{(0)}\right)}{\sqrt{l_i^{(0)} h_i^{(0)}}} \\
&= \log \frac{\max\left(s_i^{(0)}, l_i^{(1)}, h_i^{(0)}\right)}{\max\left(s_i^{(0)}, l_i^{(0)}, h_i^{(0)}\right)} - \log \frac{\sqrt{l_i^{(1)} h_i^{(0)}}}{\sqrt{l_i^{(0)} h_i^{(0)}}} \\
&\leq \left|\log \frac{l_i^{(1)}}{l_i^{(0)}}\right| + \left|\log \frac{l_i^{(0)}}{l_i^{(1)}}\right| \in O(1), \quad \forall\, j \in \mathcal{N}_i \qquad (12)
\end{aligned}$$

Finally, we show that $|\mathcal{N}_i| \in O(1)$. To that effect we solve a general problem: What is the maximum number of points in Q, whose nearest neighbor is $\mathbf{q}_1$? Without loss of generality, let $\mathbf{q}_2$ be the nearest neighbor of $\mathbf{q}_1$, such that $\|\mathbf{q}_2 - \mathbf{q}_1\| = 1$. Let $\mathbf{q}_3$'s nearest neighbor be $\mathbf{q}_1$. Consider the line segment $\overline{\mathbf{q}_1\mathbf{q}_3}$, and its intersection with $\mathcal{B}(\mathbf{q}_1, 1)$ the ball centered at $\mathbf{q}_1$, and passing through $\mathbf{q}_2$. Call the intersection point $\mathbf{q}_3'$. For any general point $\mathbf{q}_x \in Q$, we similarly define $\mathbf{q}_x'$. Note $\mathbf{q}_2' = \mathbf{q}_2$. By triangle inequality, we have $\|\mathbf{q}_3' - \mathbf{q}_2'\| \geq \|\mathbf{q}_3 - \mathbf{q}_2\| - \|\mathbf{q}_3 - \mathbf{q}_3'\| = \|\mathbf{q}_3 - \mathbf{q}_2\| - \|\mathbf{q}_3 - \mathbf{q}_1\| + \|\mathbf{q}_1 - \mathbf{q}_3'\|$. Since $\mathbf{q}_1$ is the nearest neighbor of $\mathbf{q}_3$, we have $\|\mathbf{q}_3 - \mathbf{q}_2\| \geq \|\mathbf{q}_3 - \mathbf{q}_1\|$. Therefore $\|\mathbf{q}_3' - \mathbf{q}_2'\| \geq \|\mathbf{q}_1 - \mathbf{q}_3'\| = 1$. Therefore, for any $x, y \in [n], x \neq y$, such that $\mathbf{q}_1$ is the nearest neighbor of $\mathbf{q}_x$ and $\mathbf{q}_y$, we have $\|\mathbf{q}_x' - \mathbf{q}_y'\| \geq 1$. Let $\mathcal{M}_1 = \{\mathbf{q}_x' \mid \mathbf{q}_1 \text{is the nearest neighbor of } \mathbf{q}_x\}$. Draw a ball of radius $1/2$ around $\mathbf{q}_1$, and every member of $\mathcal{M}_1$. Each of these balls are non-overlapping. However all of them are contained within a ball around $\mathbf{q}_1$ of radius $3/2$. Therefore $|\mathcal{M}_1| < (3/2)^d/(1/2)^d = 3^d \in O(1)$. Observing that $\mathcal{N}_i \leq 2\mathcal{M}_1$, and using Eq. (9), and 12 we have the result. ◀

▶ **Lemma 6** (Algorithm's contribution towards Φ). *For any iteration z, let $\varepsilon^{\langle z \rangle}$ denote the number of steps executed by the evolver. Then* TrackByZoom *increases Φ in $O\left(n + \varepsilon^{\langle z \rangle}\right)$ steps, each time by $O(1)$. In each of the remaining steps of the zth iteration, it decreases Φ by $\Theta(1)$ in an amortized sense.*

Proof. For a particular index i, we first concentrate on the zoom-out routine of our Algorithm TrackByZoom (see Line 6). Assume that the evolver leaves q_i untouched for the time being. We show later how to handle the case where the evolver moves q_i, while the algorithm is processing the index i. The algorithm increases h_i by a constant factor in Line 12. Therefore, $\Phi = \log(\max(s_i, l_i, h_i)/\sqrt{l_i h_i})$ decreases by a constant amount whenever $\max(s_i, l_i, h_i) \neq h_i$. Now suppose $\max(s_i, l_i, h_i) = h_i$ at time t, for the first time during the zoom-out routine. That implies a 2-expansion of $\mathcal{B}_i^H$ in line 12 contains the entire $\mathcal{B}_i$. Therefore, a further $1/\beta$-expansion definitely contains another X_j, $j \neq i$, satisfying the condition in line 8. We conclude that once $\max(s_i, l_i, h_i) = h_i$, the algorithm only increases Φ by a constant amount before moving on to the zoom-in routine of the algorithm (line 14).

Now, we look at the zoom-in routine of the algorithm (line 14). The algorithm only zooms in as long as the condition in line 23 is satisfied. Since we expand $\mathcal{B}_i^H$ with an expansion factor $\frac{3}{1-2\beta}$ in line 26, using Lemma 5 we conclude that $\mathcal{B}_i \subset \mathcal{B}_i^H$ in every step during the zoom-in routine. That implies $\max(s_i, l_i, h_i) = h_i$ throughout the zoom-in process, further implying $\Phi_i = \log\sqrt{h_i/l_i}$.

In line 24 we subdivide the hypothesis ball into $|\Psi| = \left(2\lceil\frac{3}{1-2\beta}\rceil\sqrt{d}\right)^d \in O(1)$ balls and make constant number of queries to the oracle. Therefore, h_i reduces by a factor of $\frac{3}{1-2\beta}/\left(2\lceil\frac{3}{1-2\beta}\rceil\right) \geq 2$. And hence, Φ reduces by at least a constant amount. We amortize the reduction over the $|\Psi|$ calls to the Oracle, to denote a constant reduction in Φ in every zoom-in step.

If the evolver moved q_i while the algorithm was processing the index i, there is a possibility that X_i might move outside of $\mathcal{B}_i^H$ during the zoom-in routine. We take care of this condition in line 16 to initiate the zoom-out process. Since the algorithm possibly increases Φ by a constant amount when switching from zoom-out to zoom-in, we conclude: For every evolver's step, our algorithm might increase Φ by at most a constant amount as well. ◀

▶ **Lemma 7** (Iteration time proportional to Potential). *There exists a constant speed-up factor σ for* TrackByZoom *such that $\Delta^{\langle z\rangle} \in \Theta_\sigma\left(n + \Phi^{\langle z\rangle}\right)$.*

Proof. By Lemma 4, the evolver can only increase the potential by $\Phi_\varepsilon^{\langle z\rangle} = O(\Delta^{\langle z\rangle})$ during the iteration. By Lemma 6 we observe that the algorithm increases Φ by some amount that is $O(n + \Delta^{\langle z\rangle})$. Since by Lemma 10, the algorithm reduces each Φ_i to at most ϕ_0, it reduces the overall potential by at most $\Phi^{\langle z\rangle} - n\phi_0 + \Phi_\varepsilon^{\langle z\rangle} + O(n + \Delta^{\langle z\rangle}) = O(n + \Phi^{\langle z\rangle} + \Delta^{\langle z\rangle})$. For a speed-up factor σ, the algorithm TrackByZoom spends $O(n + \Phi^{\langle z\rangle} + \Delta^{\langle z\rangle})/\sigma$ time to do so, implying $\Delta^{\langle z\rangle} \leq O(n + \Phi^{\langle z\rangle} + \Delta^{\langle z\rangle})/\sigma$, further implying $\Delta^{\langle z\rangle} \in O_\sigma(n + \Phi^{\langle z\rangle})$, for a sufficiently large constant σ.

To see why $\Delta^{\langle z\rangle} \in \Omega(n + \Phi^{\langle z\rangle})$, we follow a similar logic, except to observe that the evolver can decrease the potential by at most $\Phi_\varepsilon^{\langle z\rangle} = O(\Delta^{\langle z\rangle})$ as well. By Lemma 10, the algorithm reduces the overall potential Φ by at least $\Phi^{\langle z\rangle} - n\phi_0 - \Phi_\varepsilon^{\langle z\rangle} = \Omega(n + \Phi^{\langle z\rangle} - \Delta^{\langle z\rangle})$, which takes it at least $\Omega(n + \Phi^{\langle z\rangle} - \Delta^{\langle z\rangle})/\sigma$ time, implying that $\Delta^{\langle z\rangle} \geq \Omega(n + \Phi^{\langle z\rangle} - \Delta^{\langle z\rangle})/\sigma$, resulting in $\Delta^{\langle z\rangle} \in \Omega_\sigma(n + \Phi^{\langle z\rangle})$. ◀