Grouping Time-varying Data for Interactive Exploration

Arthur van Goethem[*] *Marc van Kreveld*[†] *Maarten Löffler*[†]
Bettina Speckmann[*] *Frank Staals*[‡]

Abstract

We present algorithms and data structures that support the interactive analysis of the grouping structure of one-, two-, or higher-dimensional time-varying data while varying all defining parameters. Grouping structures characterise important patterns in the temporal evaluation of sets of time-varying data. We follow Buchin et al. [9] who define groups using three parameters: group-size, group-duration, and inter-entity distance. We give upper and lower bounds on the number of maximal groups over all parameter values, and show how to compute them efficiently. Furthermore, we describe data structures that can report changes in the set of maximal groups in an output-sensitive manner. Our results hold in $\mathbb{R}^d$ for fixed d.

1 Introduction

Time-varying phenomena are ubiquitous and hence the rapid increase in available tracking, recording, and storing technologies has led to an explosive growth in time-varying data. Such data comes in various forms: time-series (tracking a one-dimensional variable such as stock prices), two- or higher-dimensional trajectories (tracking moving objects such as animals, cars, or sport players), or ensembles (sets of model runs under varying initial conditions for one-dimensional variables such as temperature or rain fall), to name a few. Efficient tools to extract information from time-varying data are needed in a variety of applications, such as predicting traffic flow [22], understanding animal movement [7], coaching sports teams [17], or forecasting the weather [25]. Consequently, recent years have seen a flurry of algorithmic methods to analyse time-varying data which can, for example, identify important geographical locations from a set of trajectories [6, 19], determine good average representations [8], or find patterns, such as groups traveling together [9, 18, 21].

Most, if not all, of these algorithms use several parameters to model the applied problem at hand. The assumption is that the domain scientists, who are the users of the algorithm, know from years of experience which parameter values to use in their analysis. However, in many cases this assumption is not valid. Domain scientists do *not* always know the correct parameter settings and in fact need algorithmic support to interactively explore their data in, for example, a visual analytics system [3, 20].

We present algorithms and data structures that support the interactive analysis of the grouping structure of one-, two-, or higher-dimensional time-varying data while varying all

[*]Department of Mathematics and Computer Science, Eindhoven University of Technology {a.i.v.goethem,b.speckmann}@tue.nl

[†]Department of Information and Computing Sciences, Utrecht University, {m.j.vankreveld, m.loffler}@uu.nl

[‡]MADALGO, Aarhus University, f.staals@cs.au.dk

defining parameters. Grouping structures (which track the formation and dissolution of groups) characterise important patterns in the temporal evaluation of sets of time-varying data. Classic examples are herds of animals or groups of people. But also for one-dimensional ensembles grouping is meaningful, for example, when detecting trends in weather models [24].

Buchin et al. [9] proposed a grouping structure for sets of moving entities. Their definition was later extended by Kostitsyna et al. [21] to geodesic distances. In this paper we use the same trajectory grouping structure. Our contributions are data structures and query algorithms that allow the parameters of the grouping structure to vary interactively and hence make it suitable for explorative analysis of sets of time-varying data. Below we first briefly review the definitions of Buchin et al. [9] and then state our contributions in detail.

Trajectory grouping structure [9]. Let $\mathcal{X}$ be a set of n entities moving in $\mathbb{R}^d$ and let $\mathbb{T}$ denote time. The entities trace trajectories in $\mathbb{T} \times \mathbb{R}^d$. We assume that each individual trajectory is piecewise linear and consists of at most τ vertices. Two entities a and b are *ε-connected* if there is a chain of entities $a = c_1, .., c_k = b$ such that for any pair of consecutive entities c_i and c_{i+1} the distance is at most ε. A set G is ε-connected, if for any pair $a, b \in G$, the entities are ε-connected (possibly using entities not in G). Given parameters m, ε, and δ, a set of entities G is an *(m, ε, δ)-group* during time interval I if (and only if) (i) G has size at least m, (ii) *duration*$(I) \geq \delta$, and (iii) G is *ε-connected* at any time $t \in I$. An (m, ε, δ)-group (G, I) is *maximal* if G is maximal in size or I is maximal in duration, that is, if there is no group $H \supset G$ that is also ε-connected during I, and no interval $J \supset I$ such that G is ε-connected during J.

Results and Organization. We want to create a data structure $\mathcal{D}$ that represents the grouping structure, that is, its maximal groups, while allowing us to efficiently change the parameters. As we show below, the complexity of the problem is already fully apparent for one-dimensional time-varying data. Hence we restrict our description to $\mathbb{R}^1$ in Sections 2–4 and then explain in Section 5 how to extend our results to higher dimensions.

If all three parameters m, ε, and δ can vary independently the question arises what constitutes a meaningful maximal group. Consider a maximal (m, ε, δ)-group (G, I). If we slightly increase ε to ε', and consider a slightly longer time interval $I' \supseteq I$ then (G, I') is a maximal $(m, \varepsilon', \delta)$-group. Intuitively, these groups (G, I) and (G, I') are the same. Thus, we are interested only in (maximal) groups that are "combinatorially different". Note that the set of entities G may also be a maximal (m, ε, δ)-group during a time interval J completely different from I, we also wish to consider (G, I) and (G, J) to be combinatorially different groups. In Section 2 we formally define when two (maximal) (m, ε, δ)-groups are (combinatorially) different. We prove that there are at most $O(|\mathcal{A}|n^2)$ such groups, where $\mathcal{A}$ is the arrangement of the trajectories in $\mathbb{T} \times \mathbb{R}^1$, and $|\mathcal{A}|$ is its complexity. We also argue that the number of maximal groups may be as large as $\Omega(\tau n^3)$, even for fixed parameters m, ε, and δ and in $\mathbb{R}^1$. This significantly strengthens the lower bound of Buchin et al. [9].

In Section 3 we present an $O(|\mathcal{A}|n^2 \log^2 n)$ time algorithm to compute all combinatorially different maximal groups. In Section 4 we describe a data structure that allows us to efficiently obtain all groups for a given set of parameter values. Furthermore we also describe data structures for the interactive exploration of the data. Specifically, given the set of maximal (m, ε, δ)-groups we want to change one or more of the parameters and efficiently report only those maximal groups which either ceased to be a maximal group or became a maximal group. That is, our data structures can answer so-called *symmetric-difference queries* which are gaining in importance as part of interactive analysis systems [16]. As mentioned above, in Section 5 we extend our data structures and algorithms to $\mathbb{R}^d$, for fixed d.

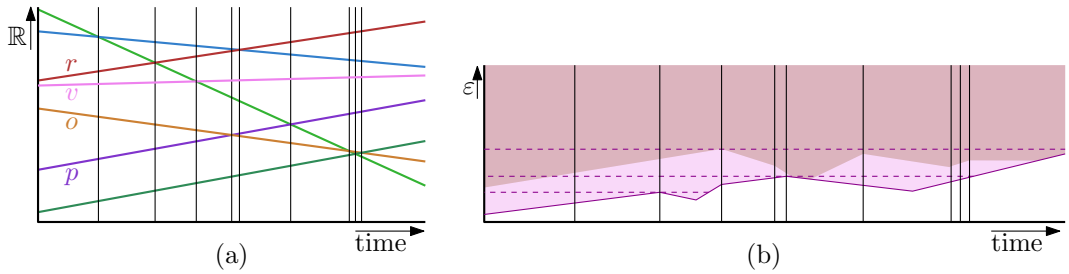


Fig. 1: (a) A set of trajectories for a set of entities moving in $\mathbb{R}^1$ (b) The region $A_{\{r,v\}}$ during which $\{r, v\}$ is alive, and its decomposition into polygons, each corresponding to a distinct instance. In all such regions, except the top one $\{r, v\}$ is a maximal group: in the top region $\{r, v\}$ is dominated by $\{r, v, o\}$ (darker region).

2 Combinatorially Different Maximal Groups

We consider entities moving in $\mathbb{R}^1$, hence the trajectories form an arrangement $\mathcal{A}$ in $\mathbb{T} \times \mathbb{R}^1$. We assume that no three pairs of entities have equal distance at the same time. Consider the four-dimensional *parameter space* $\mathbb{P}$ with axes time, size, distance, and duration. A set of entities G defines a region A_G in this space in which it is *alive*: a point $p = (p_t, p_m, p_\varepsilon, p_\delta) = (t, m, \varepsilon, \delta)$ lies in A_G if and only if G is a (m, ε, δ)-group at time t. We use these regions to define when groups are combinatorially different. First (Section 2.1) we fix $m = 1$ and $\delta = 0$ and define and count the number of combinatorially different maximal $(1, \varepsilon, 0)$-groups, over all choices of parameter ε. We then extend our results to include other values of δ and m in Section 2.2.

2.1 The Number of Distinct Maximal $(1, \varepsilon, 0)$-Groups, over all ε

Consider the (t, ε)-plane in $\mathbb{P}$ through $\delta = 0$ and $m = 1$. The intersection of all regions A_G with this plane give us the points (t, ε) for which G is a $(1, \varepsilon, 0)$-group. Note that G is a $(1, \varepsilon, 0)$-group at time t if and only if the set G is ε-connected at time t. Hence the region A_G, restricted to this plane, corresponds to the set of points (t, ε) for which G is ε-connected. A_G, restricted to this plane, is simply connected. Furthermore, as the distance between any pair of entities moving in $\mathbb{R}^1$ varies linearly, A_G is bounded from below by a t-monotone polyline f_G. The region is unbounded from above: if G is ε-connected (at time t) for some value ε, then it is also ε'-connected for any $\varepsilon' \geq \varepsilon$ (see Fig. 1). Every maximal length segment in the intersection between (the restricted) A_G and the horizontal line ℓ_ε at height ε corresponds to a (maximal) time interval I during which (G, I) is a $(1, \varepsilon, 0)$-group, or an *ε-group* for short. Every such a segment corresponds to an *instance* of ε-group G.

Observation 1. *Set G is a maximal ε-group on I, iff the line segment $s_{\varepsilon,I} = \{(t, \varepsilon) \mid t \in I\}$ is a maximal length segment in A_G, and is not contained in A_H, for a supergroup $H \supset G$.*

Two instances of ε-group G may *merge*. Let v be a local maximum of f_G and $I_1 = [t_1, v_t]$ and $I_2 = [v_t, t_2]$ be two instances of group G meeting at v. At v_ε, the two instances G that are alive during $[t_1, v_t]$ and $[v_t, t_2]$ merge and we now have a single time interval $I = [t_1, t_2]$ on which G is a group. We say that I is a new instance of G, different from I_1 and I_2. We can thus decompose A_G into maximally-connected regions, each corresponding to a distinct instance of group G, using horizontal segments through the local maxima of f_G. We further split each region at the values ε where G changes between being maximal and being dominated. Let $\mathcal{P}_G$ denote the obtained set of regions in which G is maximal. Each such a region P corresponds to a *combinatorially distinct* instance on which G is a maximal group (with at least one member

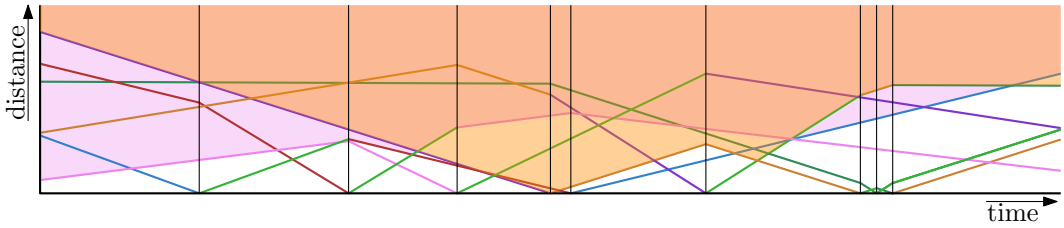


Fig. 2: The arrangement $\mathcal{H}$ and the regions $A_{\{r,v\}}$ (purple) and $A_{\{p,o\}}$ (orange) for the trajectories shown in Fig. 1(a). The arrangement $\mathcal{H}$ corresponds to the arrangement of functions $h_a(t)$ that represent the distance from a to the entity directly above a at time t.

and duration at least zero). The region P is bounded by at most two horizontal line segments and two ε-monotone chains (see Fig. 1(b)).

Counting maximal ε-groups. To bound the number of distinct maximal ε-groups, over all values of ε, we have to count the number of polygons in $\mathcal{P}_G$ over all sets G. While there are possibly exponentially many sets, there is structure in the regions A_G which we can exploit.

Consider a set of entities G and a region $P \in \mathcal{P}_G$ corresponding to a distinct instance of the maximal ε-group G. We observe that all vertices of P lie on the polyline f_G: they are either directly vertices of f_G, or they are points (t, ε) on the edges of f_G where G starts or stops being maximal. For the latter case there must be a polyline f_H, for some subgroup or supergroup of G, that intersects f_G at such a point. Furthermore, observe that any vertex (of either type) is used by at most a constant number of regions from $\mathcal{P}_G$.

Below we show that the complexity of the arrangement $\mathcal{H}$, of all polylines f_G over all G, is bounded by $O(|\mathcal{A}|n)$. Furthermore, we show that each vertex of $\mathcal{H}$ can be incident to at most $O(n)$ regions. It follows that the complexity of all polygons $P \in \mathcal{P}_G$, over all groups (sets) G, and thus also the number of such sets, is at most $O(|\mathcal{A}|n^2)$.

The complexity of $\mathcal{H}$. The *span* $S_G(t) = \{a \mid a \in \mathcal{X} \ \wedge \ a(t) \in [\min_{b\in G} b(t), \max_{b\in G} b(t))\}$ of a set of entities G at time t is the set of entities between the lowest and highest entity of G at time t (for technical reasons, we include the lowest entity of G in the span, but not the highest). Let $h_a(t)$ denote the distance from entity a to the entity directly above a at time t, that is, $h_a(t)$ is the height of the face in $\mathcal{A}$ that has a on its lower boundary at time t.

Observation 2. *A set G is ε-connected at time t, if and only if the largest distance among consecutive entities in $S_G(t)$ is at most ε. That is,*

$$f_G(t) = \max_{a \in S_G(t)} h_a(t)$$

It follows that $\mathcal{H}$ is a subset of the arrangement of the n functions h_a, for $a \in \mathcal{X}$ (see Fig. 2). We use this fact to show that $\mathcal{H}$ has complexity at most $O(|\mathcal{A}|n)$:

Lemma 3. *Let $\mathcal{A}$ be an arrangement of n line segments, and let k be the maximum number of line segments intersected by a vertical line. The number of triplets (F, F', x) such that the faces $F \in \mathcal{A}$ and $F' \in \mathcal{A}$ have equal height h at x-coordinate x is at most $O(|\mathcal{A}|k) \subseteq O(|\mathcal{A}|n) \subseteq O(n^3)$.*

Proof. Let ℓ_x be the vertical line through point $(x, 0)$. Now consider a triplet (F, F', x), and let e_F and f_F ($e_{F'}$ and $f_{F'}$) be the two edges of F (F') intersected by ℓ_x. We charge (F, F', x) to edge $e \in \{e_F, f_F, e_{F'}, f_{F'}\}$ if (and only if) its left endpoint, say u, is the rightmost endpoint that lies to the left of ℓ_x (i.e. u is the rightmost among the left endpoints). We now show that each edge can be charged at most $2k$ times.

Consider an edge $e = \overline{uv}$ of $\mathcal{A}$, with $u_x \leq v_x$. Edge e is charged by a triplet (F, F', x), only if one of the faces, say F, is incident to e, and the left-endpoints of the three other edges that are

intersected by ℓ_x and bounding F or F' lie to the left of u. It now follows that there are only k choices for face F', as both the edges bounding F' are intersected (consecutively) by the vertical line through u_x, and any vertical line intersects at most k edges. Clearly, e is incident to at most two faces, and thus there are also only two choices for F. Finally, observe that for each such pair of faces F and F' there is at most one value $x \in [u_x, r]$, where r is the x-coordinate of the leftmost right endpoint among e_F, f_F, $e_{F'}$ and $f_{F'}$, at which F and F' have equal height (as the height of faces F and F' varies linearly in such an interval). It follows that every edge $e \in \mathcal{A}$ is charged at most $2k$ times. □

Remark 1. *Interestingly, this bound is tight in the worst case. In Appendix A we give a construction where there are $\Omega(n^3)$ triplets (F, F', x) such that F and F' have equal height at x, even if we use lines instead of line segments.*

Lemma 4. *The arrangement $\mathcal{H}$ has complexity $O(|\mathcal{A}|n)$.*

Proof. Vertices in $\mathcal{H}$ are either (i) vertices of individual functions h_a, or (ii) intersections between two such functions, say h_a and h_c. The total complexity of the individual functions is $O(|\mathcal{A}|)$, hence there are also only $O(|\mathcal{A}|)$ vertices of the type (i). Vertices of the type (ii) correspond to a triplet (F, F', x) in which F and F' are faces of $\mathcal{A}$ that have equal height at x-coordinate x. By Lemma 3 there are at most $O(|\mathcal{A}|n)$ such triplets. Thus, the number of vertices of type (ii) is also at most $O(|\mathcal{A}|n)$. □

What remains to show is that each vertex v of $\mathcal{H}$ can be incident to at most $O(n)$ polygons from different sets. We use Lemma 5, which follows from Buchin et al. [9]:

Lemma 5. *Let $\mathcal{R}$ be the Reeb graph for a fixed value ε capturing the movement of a set of n entities moving along piecewise-linear trajectories in $\mathbb{R}^d$ (for some constant d), and let v be a vertex of $\mathcal{R}$. There are at most $O(n)$ maximal groups that start or end at v.*

Lemma 6. *Let v be a vertex of $\mathcal{H}$. Vertex v is incident to at most $O(n)$ polygons from $\mathcal{P} = \bigcup_{G \subseteq \mathcal{X}} \mathcal{P}_G$.*

Proof. Let $P \in \mathcal{P}_G$ be a region that uses v. Thus, G either starts or ends as a maximal v_ε-group at time v_t. This means, v correspond to a single vertex u in the Reeb graph, built with parameter v_ε. By Lemma 5, there are at most $O(n)$ maximal v_ε-groups that start or end at u. Hence, v can occur in regions of at most $O(n)$ different sets G. For a fixed set G, the regions in $\mathcal{P}_G$ are disjoint, so there are only $O(1)$ regions from $\mathcal{P}_G$, that contain v. □

Lemma 7. *The number of distinct ε-groups, over all values ε, and the total complexity of all regions $\mathcal{P} = \bigcup_{G \subseteq \mathcal{X}} \mathcal{P}_G$, are both at most $O(|\mathcal{H}|n) = O(|\mathcal{A}|n^2)$.*

2.2 The Number of Distinct Maximal Groups, over all Parameters

Maximal groups are monotonic in m and δ (see Buchin et al. [9]); hence a maximal (m, ε, δ)-group is also a maximal $(m', \varepsilon, \delta')$-group for any parameters $m' \leq m$ and $\delta' \leq \delta$. It follows that the number of combinatorially different maximal groups is still at most $O(|\mathcal{A}|n^2)$.

For the complexity of the regions in $\bigcup \mathcal{P}_G$: fix $m = 0$, and consider the remaining subspace of $\mathbb{P}$ with axes time, distance, and duration, and the restriction of A_G, for any set G, into this space. In the $\delta = 0$ plane we simply have the regions A_G, that are bounded from below by a t-monotone polyline f_G, as described in Section 2.1. As we increase δ we observe that the local minima in the boundary f_G get replaced by a horizontal line segment of width δ (see Fig. 3). For arbitrarily small values of $\delta > 0$, the total complexity of this boundary is still $O(|\mathcal{A}|n^2)$.

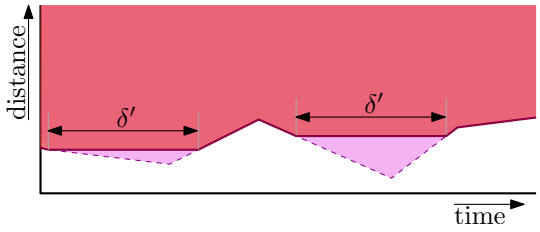


Fig. 3: A cross section of the region $A_{\{r,v\}}$ with the plane through $\delta = \delta'$. The boundary of the original region (i.e. the cross section with the plane through $\delta = 0$) is dashed.

Further increasing δ, monotonically decreases the number of vertices on the functions f_G. It follows that the regions A_G, restricted to the time, distance, duration space also have total complexity $O(|\mathcal{A}|n^2)$. Finally, consider the regions A_G in the full four dimensional space. Clearly, $A_G \cap \{p \mid p \in \mathbb{P} \wedge p_m < |G|\} = \emptyset$. For values $m \geq |G|$, the boundary of A_G is constant in m. We conclude:

Theorem 8. *Let $\mathcal{X}$ be a set of n entities, in which each entity travels along a piecewise-linear trajectory of τ edges in $\mathbb{R}^1$, and let $\mathcal{A}$ be the resulting trajectory arrangement. The number of distinct maximal groups is at most $O(|\mathcal{A}|n^2) = O(\tau n^4)$, and the total complexity of all regions in the parameter space corresponding to these groups is also $O(|\mathcal{A}|n^2) = O(\tau n^4)$.*

In Section B in the appendix we prove Lemma 9: even for fixed parameters ε, m, and δ, the number of maximal (m, ε, δ)-groups, for entities moving in $\mathbb{R}^1$, may be as large as $\Omega(\tau n^3)$. This strengthens the result of Buchin et al. [9], who established this bound for entities in $\mathbb{R}^2$.

Lemma 9. *For a set $\mathcal{X}$ of n entities, in which each entity travels along a piecewise-linear trajectory of τ edges in $\mathbb{R}^1$, there can be $\Omega(\tau n^3)$ maximal ε-groups.*

3 Algorithm

In the following we refer to combinatorially different maximal groups simply as groups. Our algorithm computes a representation (of size $O(|\mathcal{A}|n^2)$) of all groups, which we can use to list all groups and, given a pointer to a group G, list all its members and the polygon $Q_G \in \mathcal{P}_G$. We assume $\delta = 0$ and $m = 1$, since the sets of maximal groups for $\delta > 0$ and $m > 1$ are a subset of the set for $\delta = 0$ and $m = 1$.

3.1 Overview

Our algorithm uses the arrangement $\mathcal{H}$ located in the (t, ε)-plane. Line segments in $\mathcal{H}$ correspond to the height function of the faces in $\mathcal{A}$. Let $a, b \in S_G(t)$ be the pair of consecutive entities in the span of a group G with maximum vertical distance at time t. We refer to (a, b) as the *critical pair* of G at time t. The pair (a, b) determines the minimal value of ε that is required for the group G to be ε-connected at time t. The distance between a critical pair (a, b) defines an edge of the polygon bounding G in $\mathcal{H}$.

Our representation will consist of the arrangement $\mathcal{H}$ in which each edge e is annotated with a data structure $\mathcal{T}_e$, a list $\mathcal{L}$ (or array) with the top edge in each *group polygon* $Q_G \in \mathcal{P}_G$, and an additional data structure $\mathcal{S}$ to support reconstructing the grouping polygons. We start by computing the arrangement $\mathcal{H}$. This takes $O(|\mathcal{H}|) = O(\tau n^3)$ time [2]. The arrangement is built from the set of height-functions of the faces of $\mathcal{A}$. With each edge we store the pair of edges in $\mathcal{A}$ responsible for it.

Given arrangement $\mathcal{H}$ we use a sweep line algorithm to construct the rest of the representation. A horizontal line $\ell(\varepsilon)$ is swept at height ε upwards, and all groups G whose group polygon Q_G currently intersects ℓ are maintained. To achieve this we maintain a two-part status structure. First, a set $\mathcal{S}$ with for each group G the time interval $I(G, \varepsilon) = Q_G \cap \ell(\varepsilon)$. Second, for each edge $e \in \mathcal{H}$ intersected by $\ell(\varepsilon)$ a data structure $\mathcal{T}_e$ with the sets of entities whose time interval starts or ends at e, that is, $G \in \mathcal{T}_e$ if and only if $I(G, \varepsilon) = [s, t]$ with $s = e \cap \ell(\varepsilon)$ or $t = e \cap \ell(\varepsilon)$. We postpone the implementation of $\mathcal{T}$ to Section 3.3. The data structures support the following operations:

Operation	Input	Action
FILTER$(\mathcal{T}_e, X)$	A data structure $\mathcal{T}_e$ A set of entities X	Create a data structure $\mathcal{T}' = \{G \cap X \mid G \in \mathcal{T}_e\}$
INSERT$(\mathcal{T}_e, G)$	A data structure $\mathcal{T}_e$ A pointer to a representation of G	Create a data structure $\mathcal{T}' = \mathcal{T}_e \cup \{G\}$.
DELETE$(\mathcal{T}_e, G)$	A data structure $\mathcal{T}_e$ A pointer to a representation of G	Create a data structure $\mathcal{T}' = \mathcal{T}_e \setminus \{G\}$.
MERGE$(\mathcal{T}_e, \mathcal{T}_f)$	Two data structures $\mathcal{T}_e$, $\mathcal{T}_f$, belonging to two edges e, f having the same starting or ending vertex	Create a data structure $\mathcal{T}' = \mathcal{T}_e \cup \mathcal{T}_f$.
CONTAINS$(\mathcal{T}_e, G)$	A data structure $\mathcal{T}_e$ A pointer to a representation of G ending or starting on edge e	Test if $\mathcal{T}_e$ contains set G.
HASSUPERSET$(\mathcal{T}_e, G)$	A data structure $\mathcal{T}_e$ A pointer to a representation of G ending or starting on edge e	Test if $\mathcal{T}_e$ contains a set $H \supseteq G$, and return the smallest such set if so.

The end points of the time interval $I(G, \varepsilon) = [\mathit{start}(G, \varepsilon), \mathit{end}(G, \varepsilon)]$ vary non-stop along the sweep. For each group G, the set $\mathcal{S}$ instead stores the edges e and f of $\mathcal{H}$ that contain the starting time $\mathit{start}(G, \varepsilon)$ and ending time $\mathit{end}(G, \varepsilon)$, respectively, and pointers to the representation of G in $\mathcal{T}_e$ and $\mathcal{T}_f$. We refer to e and f as the *starting edge* and *ending edge* of G. In addition, we store with each interval $I(G, \varepsilon)$ a pointer to the previous version of the interval $I(G, \varepsilon')$ if (and only if) the starting time (ending time) of G changed to edge e (edge f) at ε'. Note that updates for both $\mathcal{S}$ and $\mathcal{T}$ occur only when a vertex is hit by the sweep line $\ell(\varepsilon)$. For all unbounded groups we add $I(G, \infty)$ to $\mathcal{L}$ after the sweep line algorithm.

3.2 Sweepline Events

The sweep line algorithm results in four different vertex events (see Fig. 4). The EXTEND-event has a symmetrical version in which $\overline{uu'}$ and $\overline{ww'}$ both have a negative incline. We describe how to update our hypothetical data structures in all cases.

Case I - Birth. Vertex v is a local minimum of one of the functions h_a, with $a \in \mathcal{X}$ (see Fig. 4(a)). When the sweep line intersects v a new maximal group G is born. We can find the maximal group spawned in $O(|G|)$ time by checking which trajectories are ε-connected for this value of t and ε. To this end we traverse the (vertical decomposition of) $\mathcal{A}$ starting at the entities defining v.

Case II - Extend. Vertex v is the intersection of two line segments $s_{ab} = \overline{uu'}$ and $s_{cd} = \overline{ww'}$, both with a positive incline (see Fig. 4(b)). The case in which s_{ab} and s_{cd} have negative incline can be handled symmetrically. Assume without loss of generality that s_{cd} is steeper than s_{ab}. We start with the following observation:

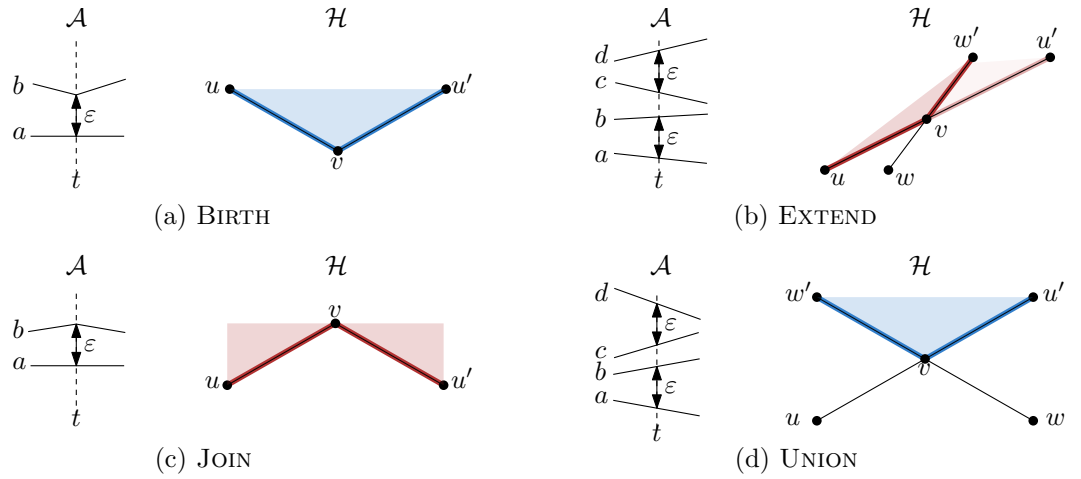


Fig. 4: The different types of vertex events shown both in the arrangement $\mathcal{A}$ and in $\mathcal{H}$. The Extend event has a horizontally symmetric case.

Observation 10. *None of the groups arriving on edge (w, v) continue on edge (v, u').*

Proof. Let G be a group that arrives at v using edge (w, v). As G uses (w, v), it must contain entities both above and below the face F defined by critical pair (c, d). We know that u_ε and w_ε are strictly smaller than v_ε and ε is never smaller than zero. Thus, v_ε is strictly positive and F has a strictly positive height at t. Therefore, G still contains entities above and below F after time t. But then the critical pair (c, d) is still part of G and s_{cd} is a lower bound for the group. It follows that G must use edge (v, w'). □

We first compute the groups on outgoing edge (v, u'). By Observation 10 all these groups arrive on edge (u, v). In particular, they are the maximal size subsets from $\mathcal{T}_{(u,v)}$ for which all entities lie below entity d at time t, that is, $\mathcal{T}_{(v,u')} = \textsc{Filter}(\mathcal{T}_{(u,v)}, \mathit{below}(d, t))$, where $\mathit{below}(y, t) = \{x \mid x \in \mathcal{X} \wedge x(t) < y(t)\}$. For each group G in $\mathcal{T}_{(v,u')}$ we update the time-interval in $\mathcal{S}$. If G was dominated by a maximal group $H \supset G$ on incoming edge (u, v), we insert a new time interval with starting edge $f = \mathit{start}(H, \varepsilon)$ and ending edge (v, u') into $\mathcal{S}$, and insert G into $\mathcal{T}_f$. Note that G and H indeed have the same starting time: G is a subset of H, and is thus ε-connected at any time where H is ε-connected. Since G was not maximal before, it did not start earlier than H either.

The groups from $\mathcal{T}_{(u,v)}$ that contain entities on both sides of critical pair (c, d), continue onto edge (v, w'). Let $\mathcal{T}'$ denote these groups. We update the interval $I(G)$ in $\mathcal{S}$ for each group $G \in \mathcal{T}'$ by setting the ending edge to (v, w').

Next, we determine which groups from $\mathcal{T}_{(w,v)}$ die at v. A maximal group $G \in \mathcal{T}_{(w,v)}$ dies at v if there is a group H on (v, w') that dominates G. Any such group H must arrive at v by edge (u, v). Hence, for each group $G \in \mathcal{T}_{(w,v)}$ we check if there is a group $H \in \mathcal{T}'$ with $H \supset G$ and $I(H) \supseteq I(G)$. For each of these groups we remove the interval $I(G, \varepsilon)$ from $\mathcal{S}$, add $I(G, \varepsilon)$ to $\mathcal{L}$, and delete the set G from the data structure $\mathcal{T}_f$, where f is the starting edge of G (at height ε).

The remaining (not dominated) groups from $\mathcal{T}_{(w,v)}$ continue onto edge (v, w'). Let $\mathcal{T}''$ denote this set. We obtain $\mathcal{T}_{(v,w')}$ by merging $\mathcal{T}'$ and $\mathcal{T}''$, that is, $\mathcal{T}_{(v,w')} = \textsc{Merge}(\mathcal{T}', \mathcal{T}'')$. Since we now have the data structures $\mathcal{T}_{(v,u')}$ and $\mathcal{T}_{(v,w')}$, and we updated $\mathcal{S}$ accordingly, our status structure again reflects the maximal groups currently intersected by the sweep line.

Case III - Join. Vertex v is a local maximum of one of the functions h_a, with $a \in \mathcal{X}$ (see Fig. 4(c)). Two combinatorially different maximal groups G_u and G_w with the same set of entities

die at v (and get replaced by a new maximal group G^*) if and only if G_u is a maximal group in $\mathcal{T}_{(u,v)}$ and G_w is a maximal group in $\mathcal{T}_{(w,v)}$. We test this with a call to CONTAINS$(\mathcal{T}_{(w,v)}, G_u)$ for each group $G_u \in \mathcal{T}_{(u,v)}$. Let G be a group in $\mathcal{T}_{(u,v)}$, and let $H \in \mathcal{T}_{(w,v)}$ be the smallest supergroup of G, if such a group exists. At v the group G will immediately extend to the ending edge of H. We can find H by using a HASSUPERSET$(\mathcal{T}_{(w,v)}, G)$ call. If H exists we insert G into $\mathcal{T}_e$, and update $I(G, \varepsilon)$ in $\mathcal{S}$ accordingly. We process the groups G in $\mathcal{T}_{(w,v)}$ that have a group $H \in \mathcal{T}_{(u,v)}$ whose starting time jumps at v analogously.

Case IV - Union. Vertex v is the intersection of a line segment $s_{ab} = \overline{uu'}$ with positive incline and a line segment $s_{cd} = \overline{ww'}$, with negative incline (see Fig. 4(d)). The UNION event is a special case of the BIRTH event. Incoming groups on edge (u, v) are below the line segment s_{cd} and, hence, can not contain any elements that are above c. As a consequence the line segment s_{cd} does not limit these groups and for a group $G \in \mathcal{T}_{(u,v)}$ we can safely add it to $\mathcal{T}_{(v,u')}$. We also update the interval $I(G)$ in $\mathcal{S}$ by setting the ending edge to (v, u'). An analogous argument can be made for groups arriving on edge (w, v).

Furthermore a new maximal group is formed. Let H be the set of all entities ε-connected to entity a at time t. We insert H into $\mathcal{T}_{(v,u')}$ and $\mathcal{T}_{(v,w')}$ and we insert a time interval $I(H)$ into $\mathcal{S}$ with starting edge (v, w') and ending edge (v, u').

3.3 Data Structure

We can implement $\mathcal{S}$ using any standard balanced binary search tree, the only requirement is that, given a (representation of) set G in a data structure $\mathcal{T}_e$, we can efficiently find its corresponding interval in $\mathcal{S}$.

The data structure $\mathcal{T}_e$. We need a data structure $\mathcal{T} = \mathcal{T}_e$ that supports FILTER, INSERT, DELETE, MERGE, CONTAINS, and HASSUPERSET efficiently. We describe a structure of size $O(n)$, that supports CONTAINS and HASSUPERSET in $O(\log n)$ time, FILTER in $O(n)$ time, and INSERT and DELETE in amortized $O(\log^2 n)$ time. In general, answering CONTAINS and HASSUPERSET queries in a dynamic setting is hard and may require $O(n^2)$ space [27].

Lemma 11. *Let G and H be two non-empty ε-groups that both end at time t. We have:*

$$(G \cap H \neq \emptyset \wedge |G| \leq |H|) \iff G \subseteq H \wedge G \neq \emptyset.$$

Proof. The if-direction is easy: $G \subseteq H$ immediately implies that $|G| \leq |H|$, and since G is non-empty we then also have $G \cap H = G \neq \emptyset$.

We prove the only-if direction by contradiction: assume by contradiction that $G \not\subseteq H$, and thus there is an element $b \in G \setminus H$. Furthermore, let $a \in G \cap H$, and let s_G and s_H denote the starting times of group G and H, respectively. We distinguish two cases: $s_G \leq s_H$ and $s_H < s_G$.

Case $s_G \leq s_H$. Since $a \in G$ and $s_G \leq s_H$, we have that at any time in $[s_H, z]$, the entities in G are ε-connected to a. So, in particular, entity b is ε-connected to a. However, during $[s_H, z]$, the entities in H are also ε-connected to a. Thus, it follows that during $[s_H, z]$, entity b is also ε-connected to H, and thus $b \in H$. Contradiction.

Case $s_H < s_G$. Analogous to the previous case we get that both H and G are ε-connected to entity a during $[s_G, z]$. It then follows that $H \subseteq G$. However, as $b \in G \setminus H$, this relation is strict, that is, $H \subset G$. This contradicts that $|G| \leq |H|$. □

We implement $\mathcal{T}$ with a tree similar to the *grouping-tree* used by Buchin et al. [9]. Let $\{G_1, .., G_k\}$ denote the groups stored in $\mathcal{T}$, and let $\mathcal{X}' = \bigcup_{i \in [1,..,k]} G_i$ denote the entities in these groups. Our tree $\mathcal{T}$ has a leaf for every entity in $\mathcal{X}'$. Each group G_i is represented by an internal

node v_i. For each internal node v_i the set of leaves in the subtree rooted at v_i corresponds exactly to the entities in G_i. By Lemma 11 these sets indeed form a tree. With each node v_i, we store the size of the group G_i, and (a pointer to) an arbitrary entity in G_i. Next to the tree we store an array containing for each entity a pointer to the leaf in the tree that represents it (or NIL if the entity does not occur in any group). We preprocess $\mathcal{T}$ in $O(n)$ time to support level-ancestor (LA) queries as well as lowest common ancestor (LCA) queries, using the methods of Bender and Farach-Colton [4, 5]. Both methods work only for *static* trees, whereas we need to allow updates to $\mathcal{T}$ as well. However, as we need to query $\mathcal{T}_e$ only when processing the upper end vertex of e, we can be lazy in updating $\mathcal{T}_e$. More specifically, we delay all updates, and simply rebuild $\mathcal{T}_e$ when we handle its upper end vertex.

HasSuperSet and Contains queries. Using LA queries we can do a binary search on the ancestors of a given node. This allows us to implement both HASSUPERSET$(\mathcal{T}_e, G)$ queries and CONTAINS$(\mathcal{T}_e, G)$ in $O(\log n)$ time for a group G ending or starting on edge e. Let a be an arbitrary element from group G. If the datastructure $\mathcal{T}_e$ contains a node matching the elements in G then it must be an ancestor of the leaf containing a in $\mathcal{T}$. That is, it is the ancestor that has exactly $|G|$ elements. By Lemma 11 there is at most one such node. As ancestors only get more elements as we move up the tree, we find this node in $O(\log n)$ time by binary search. Similarly, we can implement the HASSUPERSET function in $O(\log n)$ time.

Insert, Delete, and Merge queries. The INSERT, DELETE, and MERGE operations on $\mathcal{T}_e$ are performed lazily; We execute them only when we get to the upper vertex of edge e. At such a time we may have to process a batch of $O(n)$ such operations. We now show that we can handle such a batch in $O(n \log^2 n)$ time.

Lemma 12. *Let $G_1,..,G_m$ be maximal ε-groups, ordered by decreasing size, such that: (i) all groups end at time t, (ii) $G_1 \supseteq G_i$, for all i, (iii) the largest group G_1 has size s, and (iv) the smallest group has size $|G_m| > s/2$. We then have that $G_i \supseteq G_{i+1}$ for all $i \in [1,..,m-1]$.*

Proof. All groups G_i are subsets of G_1 and have size at least $s/2$. Thus, any two subsets G_i and G_{i+j} have a non-empty intersection, i.e. $G_i \cap G_{i+j} \neq \emptyset$. The result then follows directly from Lemma 11. □

Lemma 13. *Given two nodes $v_G \in \mathcal{T}$ and $v_H \in \mathcal{T}'$, representing the set G respectively H, both ending at time t, we can test if $G \subseteq H$ in $O(1)$ time.*

Proof. Let a be the entity from G stored with v_G. We use the array of $\mathcal{T}'$ to find the leaf ℓ in $\mathcal{T}'$ that represents a, and perform a LCA query on ℓ and v_H in $\mathcal{T}'$. If the result is v_H then $a \in H$ and Lemma 11 states that $G \subseteq H$ if and only if $|G| < |H|$. If the result is not v_H then $a \notin H$, and trivially $G \not\subseteq H$. Finding ℓ, and performing the LCA query takes $O(1)$ time. As we store the group size with each node, we can also test if $|G| < |H|$ in constant time. □

Lemma 14. *Given $m = O(n)$ nodes representing maximal ε-groups $G_1,..,G_m$, possibly in different data structures $\mathcal{T}_1,..,\mathcal{T}_m$, that all share ending time t, we can construct a new data structure $\mathcal{T}$ representing $G_1,..,G_m$ in $O(n \log^2 n)$ time.*

Proof. Sort the groups $G_1,..,G_m$ on decreasing group size. Let $G_1 \in \mathcal{T}_1$ denote the largest group and let it have size s. We assume for now that G_1 is a superset of all other groups. If this is not the case we add a dummy group G_0 containing all elements. We process the groups in order of decreasing size. By Lemma 12 it follows that all groups $G_1,..,G_k$ that are larger than $s/2$ form a path P in $\mathcal{T}$, rooted at G.

For all remaining (small) groups G_i we then find the smallest group in P that is a super set of G_i. By Lemma 13, we can test in $O(1)$ time if a group $H \in P$ is a supergroup of G_i by performing a LCA query in the tree H originated from. We can then find the smallest super set of G_i in $O(\log n)$ time using a binary search. Once all groups are partitioned into clusters with the same ancestor G_i, we process the clusters recursively. When the largest group in a cluster has size one we are done (see Fig. 5).

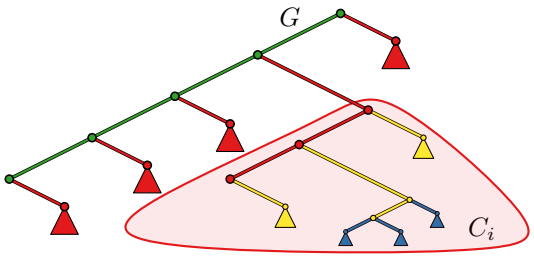


Fig. 5: $\mathcal{T}$ is built top-down in several rounds. Edges and nodes are colored by round.

The algorithm goes through a series of rounds. In each round the remaining clusters are handled recursively. Because all (unhandled) clusters jointly contain no more than $O(n)$ groups, each round takes only $O(n \log n)$ time in total. As in each round the size of the largest group left is reduced by half, it follows that after $O(\log n)$ rounds the algorithm must has constructed the complete tree. Updating the array with pointers to the leaves takes $O(n)$ time, as does rebuilding the tree for future LA and LCA queries. □

The final function FILTER can easily be implemented in linear time by pruning the tree from the bottom up. We thus conclude:

Lemma 15. *We can handle each event in $O(n \log^2 n)$ time.*

3.4 Maximal Groups

Reconstructing the grouping polygons. Given a group G, represented by a pointer to the top edge of Q_G in $\mathcal{L}$, we can construct the complete group polygon Q_G in $O(|Q_G|)$ time, and list all group members of G in $O(|G|)$ time. We have access to the top edge of Q_G. This is an interval $I(G, \hat{\varepsilon})$ in $\mathcal{S}$, specifically, the version corresponding to $\hat{\varepsilon}$, where $\hat{\varepsilon}$ is the value at which G dies as a maximal group. We then follow the pointers to the previous versions of $I(G, \cdot)$ to construct the left and right chains of Q_G. When we encounter the value $\check{\varepsilon}$ at which G is born, these chains either meet at the same vertex, or we add the final bottom edge of Q_G connecting them. To report the group members of G, we follow the pointer to $I(G, \hat{\varepsilon})$ in $\mathcal{S}$. This interval stores a pointer to its starting edge e, and to a subtree in $\mathcal{T}_e$ of which the leaves represent the entities in G.

Analysis. The list $\mathcal{L}$ contains $O(g) = O(|\mathcal{A}|n^2)$ entries (Theorem 8), each of constant size. The total size of all $\mathcal{S}$'s is $O(|\mathcal{H}|n)$: at each vertex of $\mathcal{H}$, there are only a linear number of changes in the intervals in $\mathcal{S}$. Each edge e of $\mathcal{H}$ stores a data structure $\mathcal{T}_e$ of size $O(n)$. It follows that our representation uses a total of $O(|\mathcal{H}|n) = O(|\mathcal{A}|n^2)$ space. Handling each of the $O(|\mathcal{H}|)$ nodes requires $O(n \log^2 n)$ time, so the total running time is $O(|\mathcal{A}|n^2 \log^2 n)$.

Theorem 16. *Given a set $\mathcal{X}$ of n entities, in which each entity travels along a trajectory of τ edges, we can compute a representation of all $g = O(|\mathcal{A}|n^2)$ combinatorial maximal groups $\mathcal{G}$ such that for each group in $\mathcal{G}$ we can report its grouping polygon and its members in time linear in its complexity and size, respectively. The representation has size $O(|\mathcal{A}|n^2)$ and takes $O(|\mathcal{A}|n^2 \log^2 n)$ time to compute, where $|\mathcal{A}| = O(\tau n^2)$ is the complexity of the trajectory arrangement.*

4 Data Structures for Maximal Group Queries

In this section we present data structures that allow us to efficiently obtain all groups for a given set of parameter values (Section 4.1), and for the interactive exploration of the data (Section 4.2).

Throughout this section, n denotes the number of entities considered, τ the number of vertices in any trajectory, k the output complexity, i.e. the number of groups reported, g the number of maximal groups, g' the maximum number of maximal groups for a given (fixed) value of ε, and Π the total complexity of the regions corresponding to the g combinatorially different maximal groups. So we have $g' = O(\tau n^3)$ and $g \leq \Pi = O(\tau n^4)$. When g', g, or Π appear as the argument of a logarithm, we write $O(\log n\tau)$.

4.1 Quering the maximal groups

We show that we can store all groups in a data structure of size $O(\Pi \log n\tau \log n)$ that can be built in $O(\Pi \log^2 n\tau \log n)$ time, and allows reporting all (m, ε, δ)-groups in $O(\log^2 n\tau \log n + k)$ time. We use the following three-level tree to achieve this.

On the first level we have a balanced binary tree with in the leaves the group sizes $1...n$. Each internal node v corresponds to a range R_v of group sizes and stores all groups whose size lies in the range R_v. Let $\mathcal{G}_v$ denote this set of groups, and for each such group let D_G denote the duration of group G as a function of ε. The functions D_G are piecewise-linear, δ-monotone, and may intersect (see Fig. 6). By Theorem 8 the total complexity of these functions is $O(\Pi)$. We store all functions D_G, with $G \in \mathcal{G}_v$, in a data structure that can answer the following *polyline stabbing* queries in $O(\log^2 n\tau + k)$ time: Given a query point $q = (\varepsilon, \delta)$, report all polylines that pass above point q, that is, for which $D_G(\varepsilon) \geq \delta$. Thus, given parameters m, ε, and δ, finding all (m, ε, δ)-groups takes $O(\log^2 n\tau \log n + k)$ time.

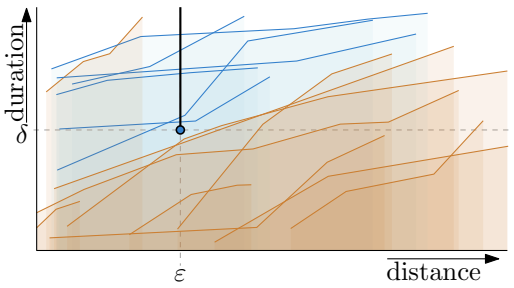


Fig. 6: The functions D_G expressing the duration of group G as a function of ε. Assuming all groups have size at least m, all (m, ε, δ)-groups intersect the upward vertical half-ray starting in point (ε, δ).

We build a segment tree storing the (ε-extent of the) individual edges of all polylines stored at v. An internal node u of the segment tree corresponds to an interval $I(u)$, and stores the set of edges $Ints(u)$ that completely span $I(u)$. Hence, with respect to u, we can consider these segments as lines. For a query with a point q, we have to be able to report all (possibly intersecting) lines from $Ints(u)$ that pass above q. We use a duality transform to map each line ℓ to a point ℓ^* and query point q to a line q^*. The problem is then to report all points ℓ^* in the half-plane below q^*. Such queries can be answered in $O(\log h + k)$ time, using $O(h)$ space and $O(h \log h)$ preprocessing time, where h is the number of points stored [13]. It follows that we can find all k polylines that pass above q in $O(\log^2 n\tau + k)$ time, using $O(\Pi \log n\tau)$ space, and $O(\Pi \log^2 n\tau)$ preprocessing time. We thus obtain the following result:

Theorem 17. *Given parameters m, ε, and δ, we can build a data structure of size $O(\Pi \log n\tau \log n)$, using $O(\Pi \log^2 n\tau \log n)$ preprocessing time, which can report all (m, ε, δ)-groups in $O(\log^2 n\tau \log n + k)$ time, where k is the output complexity.*

4.2 Symmetric Difference Queries

Here we describe data structures for the interactive exploration of the data. We often have all (m, ε, δ)-groups, for some parameters m, ε, and δ, and we want to change (some of the) parameters, say to m', ε', and δ', respectively. Thus, we need a way to efficiently report all changes in the maximal groups. This requires us to solve *symmetric difference queries*, in

which we want to efficiently report all maximal (m, ε, δ)-groups that are no longer maximal for parameters m', ε', and δ', and all maximal $(m', \varepsilon', \delta')$-groups that were not maximal for parameters m, ε, and δ. That is, we wish to report $\mathcal{G}(m, \varepsilon, \delta) \,\Delta\, \mathcal{G}(m', \varepsilon', \delta')$.

Changing only δ. Consider the case in which we vary only δ, and keep m and ε fixed, that is, $m' = m$ and $\varepsilon' = \varepsilon$. With fixed ε, it suffices to use the algorithm from Buchin et al. [9] to compute all maximal ε-groups with size at least m. There are at most g' such groups. Each group G corresponds to an interval $I_G = (-\infty, \mathit{duration}(G)]$ such that G is maximal for a choice δ of the duration parameter if and only if $\delta \in (-\infty, \mathit{duration}(G)]$. We now have two values δ and δ', and we should report all intervals in $S(\delta) \,\Delta\, S(\delta')$, where $S(x)$ denotes the intervals that contain x.

Note that we can assume without loss of generality that $\delta \leq \delta'$. Then we observe that we should report group G if and only if $\mathit{duration}(G) \in [\delta, \delta']$. Hence, our data structure is simply a balanced binary search tree on at most g' values $\mathit{duration}(G)$ and a symmetric difference query is a 1-dimensional range query.

Changing only m. The case in which we vary only m can be solved analogously to the previous case. A maximal group G has a size $|G|$, and G should be reported if and only if $|G| \in [m, m')$, assuming that the group size changes from m to m' or vice versa, with $m < m'$.

Changing only ε. The minimum duration δ is fixed, so consider the δ-truncated grouping polygons (i.e. the regions A_G where each local minimum has been replaced by a horizontal line segment of width δ). Compute all combinatorially distinct maximal groups for this parameter δ and remove all groups that have size less than m. A group G is now maximal during some interval $I_G = [\check{\varepsilon}_G, \hat{\varepsilon}_G]$, and we have to report G if (and only if) I_G occurs in the set $S(\varepsilon) \,\Delta\, S(\varepsilon')$. We now observe that this is the case exactly when I_G contains ε or ε', but not both (see Fig. 7). Using an interval tree we can thus report the symmetric difference for ε in $O(\log n\tau + k)$ time, using $O(\Pi)$ space and $O(\Pi \log n\tau)$ preprocessing time.[1]

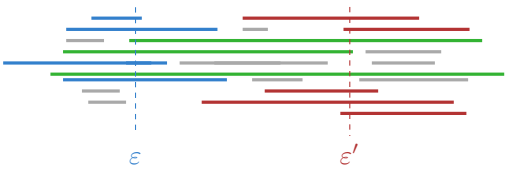


Fig. 7: The symmetric difference for parameters ε and ε' (red and blue intervals) is exactly the set of intervals that contains either ε or ε', but not both. Hence, the green intervals should not be reported.

Changing δ and m simultaneously. Consider the space $\delta \times m$. A group G is maximal in the quadrant $(-\infty, \mathit{duration}(G)] \times (-\infty, |G|]$ with top-right corner $p_G = (\mathit{duration}(G), |G|)$. So, for parameters δ and m, the set of maximal (m, ε, δ)-groups corresponds to the set of corner points that lie to the top-right of (δ, m). It now follows that when we change the parameters to (δ', m'), the maximal groups that we have to report lie in $Q_{(\delta,m)} \,\Delta\, Q_{(\delta',m')}$ (see Fig. 8). We can report those points (groups) by two three-sided (orthogonal) range queries. Therefore, we store the corner points in a priority search tree [15], and thus solve symmetric difference queries in $O(\log n\tau + k)$ time, and $O(g')$ space. Building a priority search tree takes $O(g' \log n\tau)$ time.

Changing ε and m simultaneously. Consider the space $\varepsilon \times m$. A group G is now a maximal group in a bottomless rectangle $R_G = [\check{\varepsilon}_G, \hat{\varepsilon}_G] \times (-\infty, |G|]$. See Fig. 9. Thus, for parameters ε and m the maximal groups all contain the point (ε, m). We find the groups that we have to report by combining the approaches for varying only ε and varying only m. Observe that G should be reported if (and only if) (ε, m) is in the rectangle R_G and (ε', m') is not, or vice versa. Assume we test for the former. We can solve this query problem with three very similar

[1] Note that we now have $O(\Pi)$ groups (intervals) rather than $O(g')$.

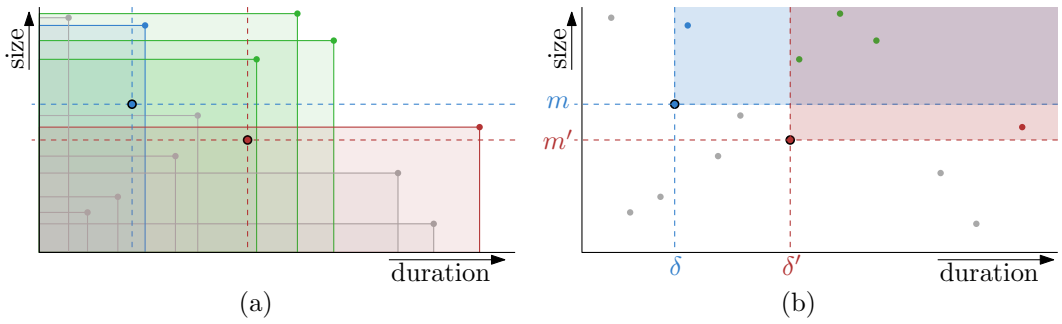


Fig. 8: Symmetric difference queries when we allow varying δ and m. Each combinatorial maximal group G corresponds to a lower-left quadrant in the space $\delta \times m$ (a). For given parameters m, ε, and m, all (δ, ε, m)-groups lie to the top-right of the point (δ, m). Therefore, the groups that have to be reported in a symmetric-difference query (shown in red and blue) can be reported by two three-sided range queries.

two-level data structures. The first is a binary search tree on all groups G sorted on $\check{\varepsilon}_G$. An internal node v is associated to a subset $\mathcal{G}_v$ of groups that appear in the subtree rooted at v. We store $\mathcal{G}_v$ by storing the horizontal line segments $[\check{\varepsilon}_G, \hat{\varepsilon}_G] \times |G|$ in a hive graph [11], preprocessed for planar point location queries. If $h = |\mathcal{G}_v|$, then this structure uses $O(h)$ storage and and allows us to report all line segments of $\mathcal{G}_v$ that lie vertically above a query point in $O(\log h + k)$ time. We query the main tree with ε' and select a subset of $O(\log n\tau)$ nodes whose associated subsets contains exactly the groups G with $\varepsilon' < \check{\varepsilon}_G$. This implies that (ε', m') is not inside R_G. The second-level structure allows us to find those groups whose rectangle R_G contains (ε, m). The second data structure is different only in its main tree, which is sorted on $\hat{\varepsilon}_G$, and we will select the nodes whose associated subsets contains exactly the groups with $\varepsilon' > \hat{\varepsilon}_G$. The third data structure is again different in the main tree only, and is sorted on $|G|$. Here we select nodes whose associated subsets have $m' > |G|$. Together, the three main trees capture that (ε', m') is not in R_G and the associated structures capture that (ε, m) is in R_G. We report any group in the symmetric difference at most twice. The data structure uses $O(\Pi \log n\tau)$ space and takes $O(\Pi \log^2 n\tau)$ time to build. The query time is $O(\log^2 n\tau + k)$.

Changing ε and δ, one by one. Consider the space $\varepsilon \times \delta$. A group G is maximal in the region below the partial, piecewise-linear, and monotonically increasing function D_G that expresses the duration of G as a function of ε. Each such partial function is defined for a single interval of ε-values. See Fig. 6. Note that the polylines representing D_G and D_H, for two groups G and H, may intersect. The combination of non-orthogonal boundaries and intersections makes changing ε and δ much harder than changing ε and m.

Consider changing parameter δ to δ', while keeping ε unchanged. For such a query we thus have to report all groups G

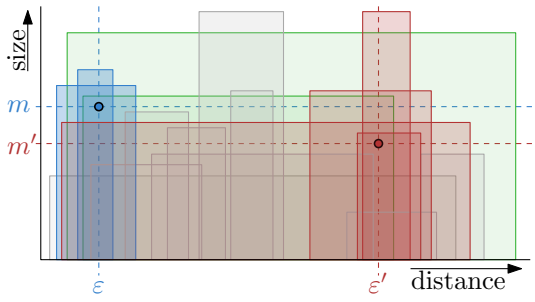


Fig. 9: Symmetric difference queries when we allow varying ε and m. Each combinatorial maximal group G now corresponds to a bottomless rectangle. We find the groups that are maximal for only one pair of parameters (i.e. the red and blue groups) by a query in a two-level tree for symmetric-difference queries.

whose polyline D_G intersects the vertical query segment $Q = \overline{(\varepsilon, \delta)(\varepsilon, \delta')}$. We use the following data structure to answer such queries. We build a segment tree storing the individual edges of the polylines. Like in Section 4, each node v in this tree now corresponds to a set $Ints(v)$ of polyline edges (one per polyline) that completely cross the interval I_v associated with v. We again treat these edges as lines. We store the $h = |Ints(v)|$ lines in a data structure by Cheng and Janardan [14] that has size $O(h \log h)$, can be built in $O(h \log^2 h)$ time, and allows reporting all (possibly intersecting) lines that intersect Q in $O(\sqrt{h} 2^{\log^* h} \log h + k)$ time. Since for any value ε there are at most $O(g')$ groups, we also have that for any node v, $|Ints(v)| = O(g')$. It follows that we can answer symmetric difference queries in δ in $O(\sqrt{g'} 2^{\log^* n\tau} \log^2 n\tau + k)$ time, after $O(\Pi \log^3 n\tau)$ preprocessing time, and using $O(\Pi \log^2 n\tau)$ space.

Consider changing parameter ε to ε', while keeping δ unchanged. For such a query we have to report all groups G whose polyline D_G is above exactly one end point of the horizontal query segment $Q = \overline{(\varepsilon, \delta)(\varepsilon', \delta)}$. Since all polylines D_G are ε and δ-monotone we could use the same approach as before, reversing the roles of ε and δ. However, a horizontal line may intersect $O(g)$ polylines rather than $O(g')$, causing $\sqrt{g}$ to appear in the query time rather than $\sqrt{g'}$. This may be significantly worse. Instead, observe that there are three ways in which Q can have exactly one end point below D_G. The two cases where one end point of Q is outside the ε-range of D_G are easily handled with a two-level tree. The first level is a binary search tree on the ε-range of D_G, and allows us to find the groups for which D_G is either defined completely before, or completely after ε. All these groups are *not* maximal for parameter ε, so among them we have to select the ones that are maximal for parameters ε' and δ'. Our second level is thus the data structure from Section 4. This leads to a data structure of size $O(\Pi \log^2 n\tau)$ and query time $O(\log^3 n\tau + k)$. The third case concerns the situation where the ε-range of Q is contained in the ε-range of D_G. In that case we need to test whether Q intersects D_G. We use a hereditary segment tree [12] on the ε-ranges of all segments S of all D_G, and at each node v, we use associated structures for the cases where segments of S are "long" and Q is "short", and vice versa. For the segments S_v of S that are long at v, we observe that there are only $O(g')$ of them, because they have a common ε-value. Furthermore, there can be at most one long segment in S_v for each group G. Hence, we can use the data structure by Cheng and Janardan [14] to report the ones intersecting Q. For the segments of S that are short at v, we know that the query segment is long and horizontal, so we can just consider the δ-span of each short segment. However, we must still ensure that the polyline D_G that a short segment is part of, extends to the right beyond Q. Both conditions together lead again to a hive graph, preprocessed for planar point location. The data structure has size $O(\Pi \log^2 n\tau)$, query time $O(\sqrt{g'} 2^{\log^* n\tau} \log^2 n\tau + k)$, and can be built in $O(\Pi \log^3 n\tau)$ time.

Changing all three parameters, one by one. To support changing all three parameters, we combine some of the previous approaches. We build two separate data structures; one to change m, the other to change ε or δ. The data structure to change m is simply the data structure from Section 4. The first level of this tree allows us to find $O(\log n)$ subtrees containing the groups whose size is in the range $(\min\{m, m'\}, \max\{m, m'\}]$. We then use the associated data structures to report the groups that are long enough (with respect to parameters ε and δ). Thus, we can answer such queries in $O(\log n \log^2 n\tau + k)$ time. To support changing ε or δ we extend the solution from the previous case: we simply add an other level to the structure, that allows us to filter the groups that intersect a query segment Q in the (ε, δ)-plane by size. This yields a query time of $O(\sqrt{g'} 2^{\log^* n\tau} \log^2 n\tau \log n + k)$. The size and preprocessing time remain unaffected, when compared to the previous situation.

Changing ε and δ simultaneously. We build a data structure that allows us to report the maximal groups for parameters ε and m as a small number of canonical subsets. For each of

these canonical subsets we store a data structure that allows us to efficiently report the groups that are *not* maximal for parameters ε' and δ'. Symmetrically, we find the groups that are not maximal for parameters ε and δ and maximal for ε' and δ'. So, basically, we need two layers of the data structure from Section 4.[2]

Recall that the data structure from Section 4 is a segment tree with associated data structures that allow half-plane range reporting. Unfortunately, the data structure that we use for the half-space range reporting does not report the result as a small number of canonical subsets. So, for the first layer of our data structure we replace these half-plane range reporting data structures by a partition tree [23]. For the second layer we can use the data structure from Section 4 as is. It follows that we can now find all groups that are maximal for ε and δ but not maximal for ε' and δ' in $O(\sqrt{g'}2^{\log^* n\tau}\log^3 n\tau + k)$ time. The data structure uses $O(\Pi \log^2 n\tau)$ space, and can be built in $O(\Pi \log^3 n\tau)$ time. We can thus solve symmetric difference queries in the same time (and with the same amount of space).

Changing all three parameters simultaneously. We use the same approach as above, expressing the groups alive for parameters m, ε, and δ as a small number of canonical subsets, for each of which we store the data structure from Section 4. It follows that we can report symmetric difference queries in $O(\sqrt{g'}2^{\log^* n\tau}\log^3 n\tau \log^2 n + k)$ time, using $O(\Pi \log^2 n\tau \log^2 n)$ space and $O(\Pi \log^3 n\tau \log^2 n)$ preprocessing time.

The following theorem summarizes our results:

Theorem 18. *Let (m, ε, δ) and $(m', \varepsilon', \delta')$ be two configurations of parameters. In $O(P(\Pi, g', n))$ time we can build a data structure of size $O(S(\Pi, g', n))$ for symmetric difference queries, that is, we can report all groups in $\mathcal{G}(\varepsilon, m, \delta) \,\Delta\, \mathcal{G}(\varepsilon', m', \delta')$, in $O(Q(\Pi, g', n, k))$ time. In these results Π denotes the total complexity of all combinatorially different maximal groups (over all values ε), g' the number of maximal groups for a fixed value ε, n the number of entities, τ the number of vertices in a trajectory, and k the output complexity. The functions P, S, and Q depend on which of the parameters are allowed to change (other parameters are assumed to be fixed and known at preprocessing time). We have*

Variable Param.	*Query time* $Q(\Pi, g', n, k)$	*Space* $S(\Pi, g', n)$	*Preproc.* $P(\Pi, g', n)$
Changing one parameter at a time			
δ	$\log n\tau + k$	$g' \log n\tau$	$g' \log n\tau$
m	$\log n\tau + k$	$g' \log n\tau$	$g' \log n\tau$
ε	$\log n\tau + k$	$\Pi \log n\tau$	$\Pi \log n\tau$
δ, m	$\log n\tau + k$	$g' \log n\tau$	$g' \log n\tau$
ε, m	$\log^2 n\tau + k$	$\Pi \log^2 n\tau$	$\Pi \log^2 n\tau$
ε, δ	$\sqrt{g'}2^{\log^* n\tau}\log^2 n\tau + k$	$\Pi \log^2 n\tau$	$\Pi \log^3 n\tau$
ε, δ, m	$\sqrt{g'}2^{\log^* n\tau}\log^2 n\tau \log n + k$	$\Pi \log^2 n\tau$	$\Pi \log^3 n\tau$
Changing multiple parameters at the same time			
δ, m	$\log n\tau + k$	$g' \log n\tau$	$g' \log n\tau$
ε, m	$\log^2 n\tau + k$	$\Pi \log^2 n\tau$	$\Pi \log^2 n\tau$
ε, δ	$\sqrt{g'}2^{\log^* n\tau}\log^3 n\tau + k$	$\Pi \log^2 n\tau$	$\Pi \log^3 n\tau$
ε, δ, m	$\sqrt{g'}2^{\log^* n\tau}\log^3 n\tau \log^2 n + k$	$\Pi \log^2 n\tau \log^2 n$	$\Pi \log^3 n\tau \log^2 n$

[2] We described this data structure for reporting all maximal groups for ε and δ. But it is easy to see that we can also use it to report all groups that are not maximal for ε and δ: we simply have to report all polylines below, rather than above, point (ε, δ).

5 Entities Moving in $\mathbb{R}^d$

We now describe how our results can be extended to entities moving in $\mathbb{R}^d$, for any constant dimension d.

5.1 Bounding the Complexity

Recall that $\mathcal{X}(t)$ denotes the locations of the entities at time t. We still consider the (t, ε)-plane in $\mathbb{P}$, and the regions A_G, for subsets $G \subseteq \mathcal{X}$, in which set G is alive. Such a region is still bounded from below by a function f_G that expresses the minimum epsilon value for which G is ε-connected. We again consider the arrangement $\mathcal{H}$ of these functions f_G, over all sets G.

Let $\mathcal{H}''$ be the arrangement of all pairwise distance functions $h_{ab}(t) = \|a(t)b(t)\|$. For any subset of entities G and any time t, $f_G(t) = \|a(t)b(t)\|$ for some pair of entities a and b. Thus, $\mathcal{H}$ is a sub-arrangement of $\mathcal{H}''$. This immediately gives an $O(\tau n^4)$ bound on the complexity of $\mathcal{H}$. We instead show that $\mathcal{H}$ has complexity at most $O(\tau n^3 \beta_4(n))$, where $\beta_s(n) = \lambda_s(n)/n$, and $\lambda_s(n)$ is the maximum length of a Davenport-Schinzel sequence of order s on n symbols. Using exactly the same argument as in Lemma 6 we then get a bound of $O(\tau n^4 \beta_4(n))$ on the number of combinatorially different groups.

Let $\mathcal{E}(t)$ be the Euclidean minimum spanning tree (EMST) of the points in $\mathcal{X}(t)$. We then observe:

Observation 19. *A subset of entities $G \subseteq \mathcal{X}$ is ε-connected at time t if and only if for any two entities $p, q \in G$ the longest edge in the Euclidean minimum spanning tree $\mathcal{E}(t)$ on the path between p and q has length at most ε.*

Specifically, let $\mathcal{E}^G(t)$ be the minimum (connected) subtree of $\mathcal{E}(t)$ containing all points in $G(t)$, and let $\check{\varepsilon}_G(t)$ be the length of the longest edge e in $\mathcal{E}^G(t)$ (see Fig. 10). We have that G is an ε-group for all $\varepsilon \geq \check{\varepsilon}_G(t)$, and that $f_G(t) = \max_{(a,b)\in\mathcal{E}^G(t)} \|a(t)b(t)\|$.

It follows from Observation 19 that we are interested in the distance function h_{ab} on the time intervals during which (a, b) is part of the EMST. Hence, we need to consider only the arrangement of such partial functions. It is, however, difficult to bound the complexity of the resulting arrangement directly. Instead, we consider h_{ab} only during those time intervals in which (a, b) is an edge in the Yao-graph [26]. Let $\mathcal{H}'$ be the resulting arrangement. Since the EMST is a subgraph of the Yao-graph it follows that $\mathcal{H}$ is a sub-arrangement of $\mathcal{H}'$ [26].

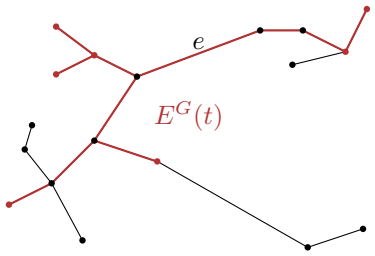


Fig. 10: $\mathcal{E}(t)$ and its minimum subtree $E^G(t)$ (red edges) for a subset of entities G (red vertices). Edge e determines the minimum ε for which G is ε-connected at t.

Lemma 20. *$\mathcal{H}'$ has complexity $O(\tau n^3 \beta_4(n))$.*

Proof. Fix an entity a, and consider the movement of the other entities with respect to a. This gives us a set of (piecewise linear) trajectories. Entity a is fixed at the origin. Partition this space into $k = O(1)$ equal size polyhedral cones $C_1, .., C_k$ that have their common apex at the origin[3]. For each such cone C_i, let $\eta_a^i(t)$ denote the distance from a to the nearest entity in the cone. It is easy to show that η_a^i is piecewise hyperbolic, and consists of $O(\tau\lambda_4(n))$ pieces [21].

Let $\mathcal{H}^*$ be the arrangement of all functions η_a^i, over all entities $a \in \mathcal{X}$ and all cones C_i. The total number of pieces (hyperbolic arcs), over all entities and all cones, is $O(\tau n \lambda_4(n))$. Partition

[3] Note that k is exponential in the dimension d.

time into $O(\tau\lambda_4(n))$ time intervals, with $O(n)$ pieces each. This may require splitting some of the pieces, but the total number of pieces remains $O(\tau n\lambda_4(n))$. In each time interval we now have $O(n)$ hyperbolic arc pieces, that intersect at most $O(n^2)$ times in total. It follows that $\mathcal{H}^*$ has complexity $O(\tau\lambda_4(n)n^2) = O(\tau n^3\beta_4(n))$.

Fix a time t, and consider the graph $Y(t)$ that has an edge (a, b) if and only if b is the nearest neighbor of a in one of the cones C_i at time t, that is, $\|a(t)b(t)\| = \eta_a^i(t)$. Indeed, $Y(t)$ is the Yao-graph of the entities at time t [26]. It follows that $\mathcal{H}^* = \mathcal{H}'$. □

Since $\mathcal{H}$ is a sub arrangement of $\mathcal{H}'$, it follows that $\mathcal{H}$ also has complexity $O(\tau n^3\beta_4(n))$. Using exactly the same argument as in Lemma 6 we then get that the number of combinatorially different maximal groups is $O(\tau n^4\beta_4(n))$. We conclude:

Theorem 21. *Let $\mathcal{X}$ be a set of n entities, in which each entity travels along a piecewise-linear trajectory of τ edges in $\mathbb{R}^d$, for any constant d. The number of maximal combinatorial groups as well as the total complexity of all their group polygons is at most $O(\tau n^4\beta_4(n))$.*

5.2 Algorithm

We can almost directly apply our algorithm from Section 3 in higher dimensions as well. Instead of the arrangement $\mathcal{H}$, we now use $\mathcal{H}'$. The only differences involve discovering the set entities involved in a Birth-event, and splitting the set of entities in case of an Extend-event. Let v denote the vertex of $\mathcal{H}'$ we are processing. We use that at time v_t, $\mathcal{H}'$ encodes the Yao-graph Y. Using a breadth first search in Y we can find the entities connected to v. If an edge has length larger than ε we stop the exploration along it. Since Y is planar, and has $O(n)$ vertices this takes $O(n)$ time. This does not affect the running time, hence we get the same result as in Theorem 16 for entities moving in $\mathbb{R}^d$, for any constant d.

5.3 Data Structures

Finding all maximal (m, ε, δ)-groups. We use the same approach as in Section 4. However, the functions $duration_G$ are no longer (piecewise) linear functions in ε. Let $start_G(\varepsilon) = f^{-1}(\varepsilon)$ and $end_G(\varepsilon) = h^{-1}(\varepsilon)$ be some hyperbolic functions f and h corresponding to curves in $\mathcal{H}$. We have that $duration_G(\varepsilon) = end_G(\varepsilon) - start_G(\varepsilon)$. The function $duration_G$ corresponds to a piecewise curve with pieces defined by polynomials of degree at most four. Hence, we have to solve the following sub-problem: given a set of g' algebraic curves of degree at most four, and query point q, report all curves that pass above q.

We can solve such queries as follows. We transform the curves into hyperplanes in $\mathbb{R}^\ell$, where ℓ is the linearization dimension. We then apply a duality transform, after which the problem can be solved using a half-space range query. Since we have curves of degree at most four in $\mathbb{R}^2$, the linearization dimension is seven: the set of points above a curve can be described using a seven-variate polynomial (the five coefficients of the degree four curve, and the two coordinates of the point) [1]. It follows that we can find all curves above query point q in $O(g'^{1-1/\lfloor 7/2\rfloor}\operatorname{polylog} n\tau + k) = O(g'^{2/3}\operatorname{polylog} n\tau + k)$ time, using linear space [23]. Reporting all maximal (m, ε, δ)-groups thus takes $O(g'^{2/3}\operatorname{polylog} n\tau + k)$ time, using $O(\Pi \log n\tau \log n)$ space and $O(\Pi \log^2 n\tau \log n)$ preprocessing time.

Alternatively, we can maintain the upper envelope of the curves in a dynamic data structure. To solve a query, we repeatedly delete the curve realizing the upper envelope at q_ε. This allows us to find all (m, ε, δ)-groups in $O(k\beta_4(g')^2\operatorname{polylog} n\tau)$ time [10].

Symmetric Difference Queries. Only the versions of the problem that involve changing both ε and δ are affected. Instead of piecewise linear functions $duration_G$ we again have piecewise curves

of degree at most four. We use a similar approach as above to find the curves that intersect a vertical or horizontal query segment in $O(g'^{2/3} \operatorname{polylog} n\tau + k)$ time. Thus, we essentially replace the $\sqrt{g'}$ terms in Theorem 18 by a $g'^{2/3}$ term.

Acknowledgments

F.S. is supported by the Danish National Research Foundation under grant nr. DNRF84. F.S. is supported by the Danish National Research Foundation under grant nr. DNRF84. A.v.G. and B.S. are supported by the Netherlands Organisation for Scientific Research (NWO) under project nr. 612.001.102 and 639.023.208, respectively.

References

[1] P. Agarwal and J. Matouek. On Range Searching with Semialgebraic Sets. *Disc. & Comput. Geom.*, 11(4):393–418, 1994.

[2] N. Amato, M. Goodrich, and E. Ramos. Computing the arrangement of curve segments: Divide-and-conquer algorithms via sampling. In *Proc. 11th ACM-SIAM Symp. on Disc. Algorithms*, pages 705–706, 2000.

[3] G. Andrienko, N. Andrienko, and S. Wrobel. Visual analytics tools for analysis of movement data. *ACM SIGKDD Explorations Newsletter*, 9(2):38–46, 2007.

[4] M. Bender and M. Farach-Colton. The LCA problem revisited. In *LATIN 2000: Theoret. Informatics*, volume 1776 of *LNCS*, pages 88–94. Springer, 2000.

[5] M. Bender and M. Farach-Colton. The level ancestor problem simplified. *Theoret. Computer Science*, 321(1):5–12, 2004.

[6] M. Benkert, B. Djordjevic, J. Gudmundsson, and T. Wolle. Finding popular places. *Int. J. of Comput. Geom. & Appl.*, 20(1):19–42, 2010.

[7] P. Bovet and S. Benhamou. Spatial analysis of animals' movements using a correlated random walk model. *J. of Theoret. Biology*, 131(4):419–433, 1988.

[8] K. Buchin, M. Buchin, M. van Kreveld, M. Löffler, R. Silveira, C. Wenk, and L. Wiratma. Median trajectories. *Algorithmica*, 66(3):595–614, 2013.

[9] K. Buchin, M. Buchin, M. van Kreveld, B. Speckmann, and F. Staals. Trajectory grouping structure. *J. of Comput. Geom.*, 6(1):75–98, 2015.

[10] T. Chan. A dynamic data structure for 3-d convex hulls and 2-d nearest neighbor queries. *J. of the ACM*, 57(3):16:1–16:15, 2010.

[11] B. Chazelle. A functional approach to data structures and its use in multidimensional searching. *SIAM J. Comput.*, 17(3):427–462, 1988.

[12] B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir. Algorithms for bichromatic line-segment problems and polyhedral terrains. *Algorithmica*, 11(2):116–132, 1994.

[13] B. Chazelle, L. Guibas, and D. Lee. The power of geometric duality. *BIT Numerical Mathematics*, 25(1):76–90, 1985.

[14] S. Cheng and R. Janardan. Algorithms for ray-shooting and intersection searching. *Journal of Algorithms*, 13(4):670–692, 1992.

[15] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, Berlin, 3rd edition, 2008.

[16] D. Eppstein, M. Goodrich, and J. Simons. Set-difference range queries. In *Proc. 2013 Canadian Conf. on Comput. Geom.*, 2013.

[17] A. Fujimura and K. Sugihara. Geometric analysis and quantitative evaluation of sport teamwork. *Systems and Computers in Japan*, 36(6):49–58, 2005.

[18] J. Gudmundsson, M. van Kreveld, and B. Speckmann. Efficient detection of patterns in 2D trajectories of moving points. *GeoInformatica*, 11:195–215, 2007.

[19] J. Gudmundsson, M. van Kreveld, and F. Staals. Algorithms for hotspot computation on trajectory data. In *Proc. 21st ACM SIGSPATIAL GIS*, pages 134–143, 2013.

[20] D. Keim, G. Andrienko, J.-D. Fekete, C. Grg, J. Kohlhammer, and G. Melanon. Visual analytics: Definition, process, and challenges. In A. Kerren, J. Stasko, J.-D. Fekete, and C. North, editors, *Information Visualization*, volume 4950 of *LNCS*, pages 154–175. Springer, 2008.

[21] I. Kostitsyna, M. van Kreveld, M. Löffler, B. Speckmann, and F. Staals. Trajectory grouping structure under geodesic distance. In *Proc. 31th Symp. Computat. Geom.* Lipics, 2015.

[22] X. Li, X. Li, D. Tang, and X. Xu. Deriving features of traffic flow around an intersection from trajectories of vehicles. In *Proc. IEEE 18th Int. Conf. Geoinformatics*, pages 1–5, 2010.

[23] J. Matouek. Efficient partition trees. *Disc. & Comput. Geom.*, 8(3):315–334, 1992.

[24] M. Mirzargar, R. Whitaker, and R. Kirby. Curve Boxplot: generalization of Boxplot for ensembles of curves. *IEEE Trans. on Vis. and Comp. Graphics*, 20(12):2654–2663, 2014.

[25] A. Stohl. Computation, accuracy and applications of trajectories – a review and bibliography. *Atmospheric Environment*, 32(6):947–966, 1998.

[26] A. Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM J. Comput.*, 11(4):721–736, 1982.

[27] D. Yellin. Representing sets with constant time equality testing. *J. of Algorithms*, 13(3):353–373, 1992.

A The Number of Equal Height Faces in an Arrangement of Lines

Recall that $\mathcal{A}$ was an arrangement of n line segments, and that $S(\mathcal{A})$ denotes the set of all triples (F, F', x) such that (i) the faces $F \in \mathcal{A}$ and $F' \in \mathcal{A}$ have equal height h at x-coordinate x, and (ii) all faces in between F and F' at x-coordinate x have height less than h. We now show that $S(\mathcal{A})$ may contain $\Omega(n^3)$ triples, even if our segments are lines.

Lemma 22. *The number of triples in $S(\mathcal{A})$ for a line arrangement $\mathcal{A}$ with n lines may be $\Omega(n^3)$.*

Proof. We construct a set of n lines $L = R \cup G \cup B \cup E$ whose arrangement $\mathcal{A}$ has $|S(\mathcal{A})| = \Omega(n^3)$. The (sub)sets R, G, and B have size k each. We use them to build the subset of faces such that there are $\Omega(k^3)$ triples (F, F', x) such that F and F' have equal height at x. The remaining $O(k)$ lines are used only to make sure that the faces in between any such pair (F, F') have smaller height. It follows that we can choose $k = \Theta(n)$ and get $|S(\mathcal{A})| = \Omega(n^3)$ as desired.

Our construction is shown in Fig. 11. The set of red lines R together with the set of blue lines B form a unit grid that has been rotated by $45 + \delta$ degrees, for some arbitrarily small $\delta > 0$. The lines in R (B) are all parallel to each other[4]. Let $C_{i,j}$ denote the face (grid cell) in which the intersection point of r_i and b_j is the point with the minimum y-coordinate, and let $\mathcal{C}_j = \{C_{1,j}, C_{2,j+1}, C_{3,j+2}.., C_{1+k_j,j+k_j}\}$ be the j^{th} "column" of such faces.

The green lines in the set $G = \{g_1,.., g_k\}$, ordered from top to bottom, are (almost) horizontally, and such that the distance between any consecutive lines g_i and g_{i+1} increases slightly (i.e. the distance between g_i and g_{i+1} is $h + i\delta'$, for some $h \in (1/c, 1)$, some constant c, and some arbitrarily small δ'). We place these green lines sufficiently far below the grid formed by the red and blue lines such that each face F_i, bounded by r_1, b_1, and the green lines g_i and g_{i+1}, is wide enough such that its x-extent contains $[x_1, x_2]$, the x-extent of the grid.

Consider the (maximal) interval J_j such that all heights of the faces (grid cells) in column j are simple increasing linear functions. See Fig. 12. It now follows that for each face $C_{1+\ell,j+\ell}$ in column j there is a small interval $I_{j,\ell} \subset J_j$ in which the height of the face varies between h and $h + k\delta'$. Hence, in this interval, the height of face $C_{1+\ell,j+\ell}$ subsequently becomes equal to the height of the faces $F_1,.., F_k$. We can choose δ' small enough such that the intervals $I_{1+\ell,j+\ell}$ for all faces in column j are disjoint (and so that $I_{1+\ell,j+\ell}$ lies to the left of $I_{2+\ell,j+\ell+1}$). So, since we have $\Omega(k)$ columns, each of $\Omega(k)$ faces, it follows that the number of x-coordinates at which two faces have equal height is $\Omega(k^3)$.

[4] Note that we can perturb all lines slightly to avoid parallel lines if desired.

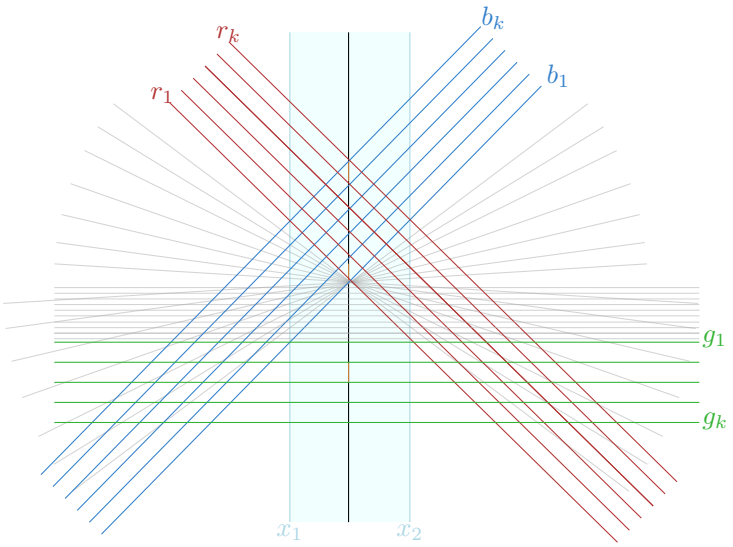


Fig. 11: A construction of n lines, such that their arrangement $\mathcal{A}$ has $|S(\mathcal{A})| = \Omega(n^3)$.

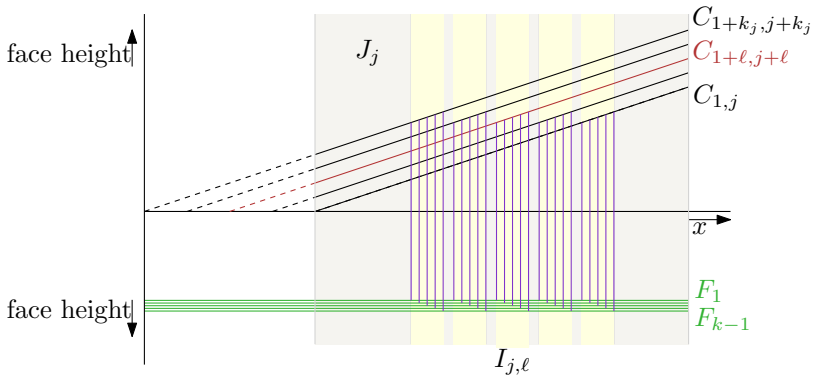


Fig. 12: The heights of the faces in a column $\mathcal{C}_j = \{C_{1,j}, C_{2,j+1}.., C_{1+k_j,j+k_j}\}$ in interval J_j.

Finally, observe that any column j, and any x-coordinate in J_j, the faces in column j, ordered from top to bottom, have decreasing height. Similarly, the faces $F_1,..,F_k$, ordered from top to bottom, have increasing height. It follows that when faces $C_{1+\ell,j+\ell}$ and F_i have equal height h', all other faces from $\mathcal{C}_\ell$ and $F_1,..,F_{i-1}$ have height smaller than h'. To make sure the remaining faces (such as the triangular face bounded by r_1, b_1, and g_1, have height at most h, we add a set of grey lines E. Since the slope of the blue lines is close to one, the distance between b_1 and g_1, at x_2 is at most $O(k)$. The same holds for the distance between r_1 and g_1 at x_1. It follows that we have to add at most $O(k)$ in between the grid and g_1 to make sure all intermediate faces have height at most h. Hence, all equal-height events involving faces $C_{1+\ell,j+\ell}$ and F_i produce a triplet in $S(\mathcal{A})$. □

B A Lower Bound on the Number of Maximal groups

In this section we show that, even for fixed parameters ε, m, and δ, the number of maximal (m,ε,δ)-groups may be as large as $\Omega(\tau n^3)$ even in $\mathbb{R}$. This strengthens the result of Buchin et al. [9], who establish this bound for entities moving in $\mathbb{R}^2$.

Lemma 9. *For a set $\mathcal{X}$ of n entities, in which each entity travels along a piecewise-linear trajectory of τ edges in $\mathbb{R}^1$, there can be $\Omega(\tau n^3)$ maximal ε-groups.*

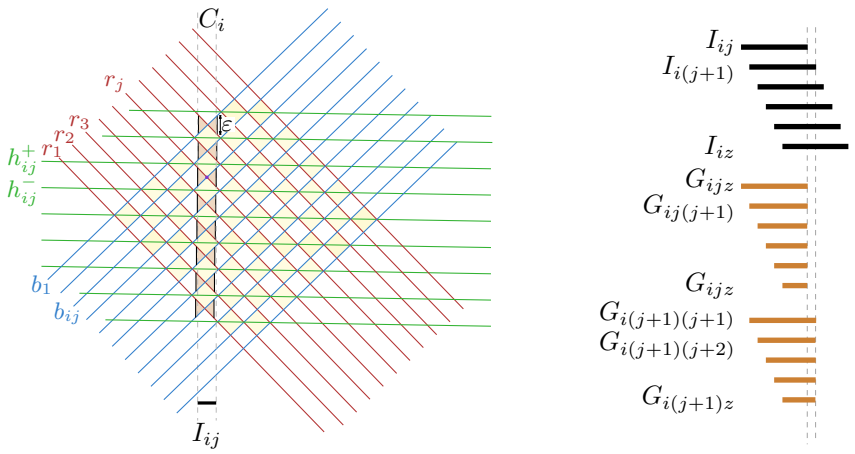
Proof. We build a construction in which all entities move along lines that yields $\Omega(n^3)$ groups. Repeating this construction $\Omega(\tau)$ times produces the claimed bound.

Partition $\mathcal{X}$ into three sets R, B, and H, with $|R| = |B| = k = \Omega(n)$ and $|H| = k/2$. The lines (forming the trajectories of the entities) in R and B form a grid, that we rotate by $45+\delta$ degrees, for some small $\delta > 0$. Let $R = \{r_1,..,r_k\}$, $B = \{b_1,..,b_k\}$, denote the lines in increasing order. See Fig. 13. We partition vertices of the rotated grid (i.e. the intersection points of R and B) into "columns" $C_1,..,C_{\Omega(n)}$, where C_i is of the form $\{(r_j,b_k),(r_{j+1},b_{k-1}),..,\}$, for some j and k. For a given column C_i, and a given line $r_j \in R$, let $b_{ij} \in B$ be the line that intersects r_j in C_i, and let t_{ij} be the time of the intersection.

Finally, for each entity r_j, j odd, we place a line $h \in H$ through the points $r_j \cap b_1$ and $r_{j+1} \cap b_2$. So, in every column C_i, h intersects some r_ℓ in $r_\ell \cap b_{i\ell}$[5]. We scale the entire construction such that the distance between two consecutive lines in h is larger than ε, and the distance between an intersection point $r_i \cap b_{ij}$ in column i and the lines h^-_{ij} and h^+_{ij} in H directly above and below it is at most ε.

Consider a column C_i, with i even. Entity r_j is directly connected to b_{ij} during some time interval $I_{ij} = [t_{ij} - \Delta, t_{ij} + \Delta]$, for some Δ. Since we rotated the grid by $45+\delta$ degrees, we have $t_{ij} < t_{i(j+1)}$

[5] Note that we can easily perturb the lines to avoid parallel lines and points in which three lines intersect.



(a) (b)

Fig. 13: (a) The construction in which the entities move along lines that yields $\Omega(n^3)$ maximal ε-groups (for a fixed ε). The background color indicates that entities are connected. (b) The intervals I_{ij} during which r_j is directly connected to b_{ij} (black), and the intervals during which the sets G_{ijz} are maximal (orange).

for every j. Hence, every interval $I_{i(j+1)}$ starts and ends slightly later than I_{ij}. Note that at any time during I_{ij}, either r_j or b_{ij} is directly connected to h_{ij}^-. The same holds for h_{ij}^+. This means that for any consecutive range $[j..z]$, the set of entities $X_{ijz} = \bigcup_{\ell\in[j..z]}\{r_\ell, b_{i\ell}, h_{i\ell}^-, h_{i\ell}^+\}$ is ε-connected during $I_{ijz} = I_{ij} \cap I_{iz} = [t_{iz} - \Delta, t_{ij} + \Delta]$. It is easy to see that I_{ijz} is maximal in duration.

It now follows that a column C_i with $\Omega(n)$ intersection points "generates" $\Omega(n^2)$ maximal ε-groups. Let $r_j,..,r_z$ be entities from R involved in column C_i. At time $t_{i\ell} - \Delta$, for $\ell \in [j..z]$, the group $G_{ij\ell}$ starts as a new maximal group. When r_j and b_{ij} disconnect, all these groups end, and the groups $G_{i(j+1)(j+1)},..,G_{i(j+1)z}$ are discovered as new maximal ε-groups. Since we have $\Omega(n)$ columns (that have $\Omega(n)$ intersection points) we get $\Omega(n^3)$ maximal ε-groups as desired. □

Note that we can choose the speed of the entities such that this construction holds for any choice of δ. Furthermore, the construction holds even for minimal group sizes of $\Omega(n)$.