Grouping Time-varying Data for Interactive Exploration

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

1 Introduction

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 (which track the formation and dissolution of groups) characterize important patterns in the evolution of sets of time-varying data. We follow Buchin et al. [4] who define groups using three parameters: group-size, group-duration, and inter-entity distance.

Trajectory grouping structure [4]. 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* at time t 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 at time t is at most ε. A set G is ε-connected, if for any pair $a, b \in G$, the entities are ε-connected. 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 Organisation. We describe a data structure $\mathcal{D}$ that represents the grouping structure, that is, its maximal groups, while allowing efficient change of the parameters. The complexity of the problem appears already in one-dimensional time-varying data. Hence we restrict our description to $\mathbb{R}^1$, the full paper extends 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". The set of entities G may also be a maximal (m, ε, δ)-group during a time interval J completely disjoint 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. [4]. In Section 3 we present an $O(|\mathcal{A}|n^2 \log^2 n)$ time algorithm to compute all combinatorially different maximal groups.

In the full paper we describe a data structure that allows us to efficiently obtain all groups for a given set of parameter values. 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 one. Our data structures can answer *symmetric-difference queries* [5].

2 Combinatorially Different Maximal Groups

We consider entities moving in $\mathbb{R}^1$, hence the trajectories form an arrangement $\mathcal{A}$ in $\mathbb{T} \times \mathbb{R}^1$. 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 which it is *alive*: a point $(t, m, \varepsilon, \delta)$ lies in A_G if and only if G is an (m, ε, δ)-group at time t. These regions help define when groups are combinatorially different. We start by fixing $m = 1$ and $\delta = 0$ to define and count the number of combinatorially different maximal $(1, \varepsilon, 0)$-groups, over all choices of parameter ε. Theorem 6 and Lemma 7 extend these results to include other values of δ and m.

Consider the (t, ε)-plane in $\mathbb{P}$ through $\delta = 0$ and $m = 1$. The intersection of all regions A_G with this plane are 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. 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,

*Department of Mathematics and Computer Science, TU Eindhoven, [a.i.v.goethem|b.speckmann]@tue.nl

†Dept. of Computing and Information Sciences, Utrecht University, The Netherlands, [m.j.vankreveld|m.loffler]@uu.nl

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

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 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 count the number of polygons in $\mathcal{P}_G$ over all sets G. Consider a distinct instance (a set of entities G and a region $P \in \mathcal{P}_G$) of the maximal ε-group G. All vertices of P lie on the polyline f_G: they are either vertices of f_G, or they are points (t, ε) on the edges of f_G where G starts or stops being maximal. Any vertex 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. 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 nearest neighbor distance among the entities in $S_G(t)$ is at most ε. Hence*

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

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

Lemma 1 *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 $O(|\mathcal{A}|k) \subseteq O(|\mathcal{A}|n) \subseteq O(n^3)$.*

Lemma 2 *The arrangement $\mathcal{H}$ has size $O(|\mathcal{A}|n)$.*

It remains to show that each vertex v of $\mathcal{H}$ can be incident to at most $O(n)$ polygons from different sets. Lemma 3 follows from Buchin et al. [4]:

Lemma 3 *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 4 *Let v be a vertex of $\mathcal{H}$. Vertex v is incident to at most $O(n)$ polygons from $\mathcal{P} = \bigcup_{G \subseteq \mathcal{X}} \mathcal{P}_G$.*

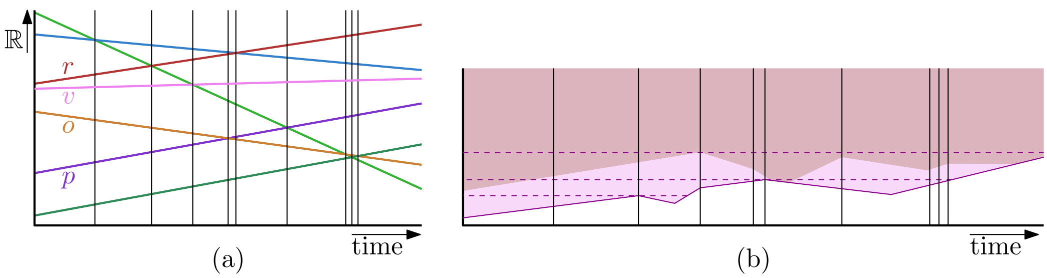

Figure 1: (a) A set of trajectories for a set of entities moving in $\mathbb{R}^1$ (b) The region $A_{\{r,v\}}$ during which $\{r, v\}$ is alive, and its decomposition into polygons, each corresponding to a distinct instance. In all such regions, except the top one $\{r, v\}$ is a maximal group: in the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

Lemma 5 *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 $O(|\mathcal{H}|n) = O(|\mathcal{A}|n^2)$.*

Theorem 6 *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 $O(|\mathcal{A}|n^2) = O(\tau n^4)$.*

Lemma 7 *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

We now 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 *grouping polygon* $Q_G \in \mathcal{P}_G$. We assume $\delta = 0$ and $m = 1$.

We use the arrangement $\mathcal{H}$ 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. The *critical pair* (a, b) determines the minimal value of ε such that the group G is ε-connected at time t. The distance between (a, b) defines an edge of the polygon bounding G in $\mathcal{H}$.

Our representation consists of the arrangement $\mathcal{H}$ in which each edge e is annotated with a data structure $\mathcal{T}_e$, a list $\mathcal{L}$ with the top edge in each grouping polygon $Q_G \in \mathcal{P}_G$, and a data structure $\mathcal{S}$ to support reconstructing the grouping polygons.

We compute $\mathcal{H}$ in $O(|\mathcal{H}|) = O(\tau n^3)$ time [1]. Given $\mathcal{H}$ we use a sweep line algorithm to construct the representation. A horizontal line $\ell(\varepsilon)$ is swept at height ε upwards, and all groups G whose grouping 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 $I(G, \varepsilon) = Q_G \cap \ell(\varepsilon)$. We can implement $\mathcal{S}$ using any standard balanced binary search tree. 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 $t = e \cap \ell(\varepsilon)$. The data structures $\mathcal{T}_e$ support the operations listed below.

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 a different edge at ε'.

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 [6].

Lemma 8 *Let G and H be two non-empty ε-groups that both end at time t. We have:*

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

We implement $\mathcal{T}$ with a tree similar to the *grouping-tree* used by Buchin et al. [4]. 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

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.

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 8 these sets indeed form a tree. With each node v_i, we store the size of G_i, and an arbitrary entity in G_i. 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 [2, 3]. Both methods work only for *static* trees, whereas we need updates to $\mathcal{T}$ as well. Since we query $\mathcal{T}_e$ only when processing the upper end vertex of e, we can be lazy in updating $\mathcal{T}_e$ and simply rebuild $\mathcal{T}_e$ when needed.

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 data structure $\mathcal{T}_e$ 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 8 there is at most one such node. As ancestors get only 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 which we can handle in $O(n \log^2 n)$ time.

Lemma 9 *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 $i \in [1,..,m-1]$.*

Lemma 10 *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.*

Lemma 11 *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.*

The final function Filter can easily be implemented in linear time by pruning the tree from the bottom up.

Lemma 12 *We can handle each event in $O(n \log^2 n)$ time.*

Reconstructing the grouping polygons. Given a group G we can construct the complete grouping 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 6), 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 13 *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 $G \in \mathcal{G}$ 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.*

References

[1] 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.

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

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

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

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

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