A Refined Definition for Groups of Moving Entities and its Computation

Marc van Kreveld[1], Maarten Löffler[*2], Frank Staals[†3], and Lionov Wiratma[‡4]

1 **Dept. of Inform. and Computing Sciences, Utrecht University, the Netherlands**
m.j.vankreveld@uu.nl

2 **Dept. of Inform. and Computing Sciences, Utrecht University, the Netherlands**
m.loffler@uu.nl

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

4 **Dept. of Inform. and Computing Sciences, Utrecht University, the Netherlands; and**
Dept. of Informatics, Parahyangan Catholic University, Indonesia
l.wiratma@uu.nl, lionov@unpar.ac.id

Abstract

One of the important tasks in the analysis of spatio-temporal data collected from moving entities is to find a *group*: a set of entities that travel together for a sufficiently long period of time. Buchin et al. [2] introduce a formal definition of groups, analyze its mathematical structure, and present efficient algorithms for computing all maximal groups in a given set of trajectories. In this paper, we refine their definition and argue that our proposed definition corresponds better to human intuition in certain cases, particularly in dense environments.

We present algorithms to compute all maximal groups from a set of moving entities according to the new definition. For a set of n moving entities in $\mathbb{R}^1$, specified by linear interpolation in a sequence of τ time stamps, we show that all maximal groups can be computed in $O(\tau^2 n^4)$ time. A similar approach applies if the time stamps of entities are not the same, at the cost of a small extra factor of $\alpha(n)$ in the running time. In higher dimensions, we can compute all maximal groups in $O(\tau^2 n^5 \log n)$ time (for any constant number of dimensions).

We also show that one τ factor can be traded for a much higher dependence on n by giving a $O(\tau n^4 2^n)$ algorithm for the same problem. Consequently, we give a linear-time algorithm when the number of entities is constant and the input size relates to the number of time stamps of each entity. Finally, we provide a construction to show that it might be difficult to develop an algorithm with polynomial dependence on n *and* linear dependence on τ.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases moving entities, trajectories, grouping, computational geometry

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2016.48

* M. L. is partially supported by the Netherlands Organisation for Scientific Research (NWO), under projects 639.021.123 and 614.001.504.
† F. S. is partially supported by the Danish National Research Foundation (grant nr. DNRF84).
‡ L. W. is supported by the Ministry of Research, Technology and Higher Education of Indonesia (138.41/E4.4/2015).

27th International Symposium on Algorithms and Computation (ISAAC 2016).
Editor: Seok-Hee Hong; Article No. 48; pp. 48:1–48:12

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Nowadays, inexpensive modern devices with advanced tracking technologies make it easy to track movements of an entity. This has led to the availability of movement data for various types of moving entities (human, animals, vehicles, etc.). Since a tracking device typically returns a single location at each time stamp, each moving entity will be represented by a moving point. Data may consist of just one trajectory tracked over a period of time, or a whole collection of trajectories that are all tracked over the same time period. Note that for the latter case, the locations of each trajectory are not necessarily collected at the same time stamps. It is common to denote the number of trajectories (or moving entities) by n and the number of time stamps used for each trajectory by τ. Hence, the input size is $\Theta(\tau n)$. Depending on the application, one of n or τ can be much larger than the other.

To analyze moving object data, a number of methods have been developed in recent times. These methods perform similarity analysis or compute a clustering, outliers, a segmentation, or various patterns that may emerge from the movement of the entities (for surveys see [3, 15]). These methods are often based on geometric algorithms, because the data is essentially spatial.

One particular type of pattern that has been well-studied is flocking [1, 4, 5]. Intuitively, a flock is a subset of the entities moving together (or simply being together) over a period of time. Other names for this and closely related concepts with slightly different definitions are herds [6], convoys [8], moving clusters [9], mobile groups [7], swarms [11], and groups [2]. Buchin et al. [2] introduce a model called the *trajectory grouping structure* which not only defines groups, but also the splitting of a group into subgroups and its opposite, merging. The algorithmic problem of reporting all maximal groups that occur in the trajectories is solved in $O(\tau n^3 + N)$ time, where $N \in O(\tau n^4)$ is the output size (the summed size of all groups reported). The algorithm also considers times in between the τ time stamps where the locations are recorded as relevant. In between these time stamps, locations are inferred by linear interpolation over time.

In this paper we continue the study of such groups, but we propose a refined definition to the one by Buchin et al. [2]. We motivate why it captures our intuition better and present algorithms to compute all maximal groups.

Previous definition of a group. The definition of a group by Buchin et al. [2] relies on three parameters: one for the distance between entities, one for the duration of a group, and one for the size of a group. We review their definitions next.

For a set of moving entities $\mathcal{X}$, two entities x and y are *directly ε-connected* at time t if the Euclidean distance between x and y is at most ε at time t, for some given $\varepsilon \geq 0$. Two entities x and y are *ε-connected in $\mathcal{X}$* at time t if there is a sequence $x = x_0, ..., x_k = y$, with $\{x_0, ..., x_k\} \subseteq \mathcal{X}$ and for all i, x_i and x_{i+1} are directly ε-connected at time t.

In [2], a *group* for an entity inter-distance ε, a minimum required duration δ, and a minimum required size m, is defined as a subset $G \subseteq \mathcal{X}$ and corresponding time interval I for which three conditions hold:

(i) G contains at least m entities.
(ii) I has a duration at least δ.
(iii) Every two entities $x, y \in G$ are ε-connected in $\mathcal{X}$ at all times in I.

Furthermore, a group G with time interval I is *maximal* if there is no time interval $I' \supset I$ for which G is also a group, and there is no proper superset $G' \supset G$ that is also a group during I [2].

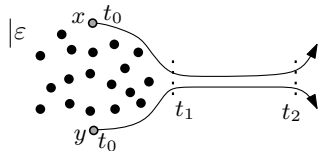

Figure 1 In the definition by [2], x and y are ε-connected during $[t_0, t_2]$.

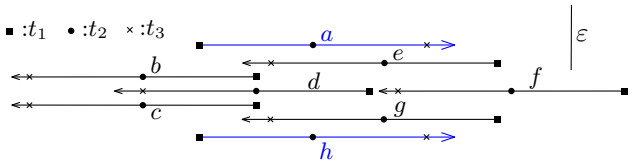

Figure 2 Entities in $G = \{a, h\}$ are ε-connected using entities not in G.

Refined definition of a group. One issue with the previous definition is that it does not correspond fully to our intuition. Two entities x and y may form a (rather small) maximal group in an interval I even if they are always far apart, as long as there are always entities of $\mathcal{X}$ in between them to make x and y ε-connected in $\mathcal{X}$. These entities in between are not part of the maximal group, but they do cause x and y to be ε-connected by the previous definition. This can have counter-intuitive effects especially in dense crowds. To avoid such issues, we refine the definition of a group. In particular, we replace condition (iii) above by:
(iii') Every two entities $x, y \in G$ are ε-connected in G during I.

We define maximal groups in the same way as before.

We give two examples that show the difference in these definitions.

First, consider a number of stationary entities S and two entities x and y, see Figure 1. Entity x starts (at time t_0) to the North of S and moves around its perimeter to the East. Entity y starts (at t_0) to the South and also moves around the perimeter to the East. After encountering (at t_1) each other at the East side, both continue together eastward, away from the stationary entities in S (ending at t_2). By the definition in [2], x and y form a maximal group in the interval $[t_0, t_2]$. By our refined definition, they form a maximal group during $[t_1, t_2]$, starting when x and y are at distance ε and actually encounter each other.

Second, the previous definition can even see groups of entities that were never close, see Figure 2. Here, $\{a, h\}$ is a maximal group in the interval $I = [t_1, t_3]$ using the definition in [2]. At each time, a and h are ε-connected, but through different subsets of entities. By choosing the coordinates carefully, we can ensure that no supergroup of $\{a, h\}$ is also a group in the same time interval, and hence $\{a, h\}$ will be maximal. Although a and h move in the same direction with the same speed, intuitively they do not form a group because they are too far apart and separated by other entities that move in the opposite direction. With our refined definition, we do not consider $\{a, h\}$ a group in the interval I, and hence also not a maximal group.

Results and Organization. We have refined the previous definition for a group of moving entities by Buchin et al. [2] and gave two examples and argue why our refined definition can give an intuitively plausible group. From now on, we will use the term "group" to denote a group of entities that comply with our refined definition.

In the following section, we show that for a set $\mathcal{X}$ of n moving entities in $\mathbb{R}^1$ with τ time stamps each, the number of maximal groups by the refined definition is $O(\tau n^3)$, which is tight in the worst case.

In Section 3, we present algorithms to compute all maximal groups in $\mathbb{R}^1$. First we consider the case where all trajectories have their vertices at the same time and begin with a basic algorithm for that runs in $O(\tau^3 n^6)$ time. Subsequent improvements lead to a running time of $O(\tau^2 n^4)$. When the time stamps of trajectories are not the same, we show that our algorithm runs in $O(\tau^2 n^4 \alpha(n))$ time.

Next, for moving entities in $\mathbb{R}^d$ $(d > 1)$, we model entities and their inter-distance into graphs and show that all maximal groups can be computed in $O(\tau^2 n^5 \log n)$ time, regardless the uniformity of the time stamps in the trajectories. We show how to achieve this bound in Section 4.

In Section 5, we consider situations where the value of n is significantly smaller than τ, which is typical in real-life moving entity datasets. We give an $O(\tau 2^n n^4)$ time algorithm for entities that move in any constant dimension.

Finally, we show an exponential bound on the number of maximal groups that can contain any given time t in the last section.

2 Preliminaries

Let $\mathcal{X}$ be a set of n entities moving in $\mathbb{R}^1$, given by locations at τ time stamps. A trajectory of an entity in $\mathcal{X}$ can be expressed by a piecewise-linear function which maps time to a point in $\mathbb{R}^1$. If $\mathbb{R}^1$ is associated with the vertical axis and time with the horizontal axis of a 2-dimensional plane, the trajectories of entities in $\mathcal{X}$ are polylines with τ vertices each. We will use the same notation to denote an entity and its trajectory. We assume that there are no two parallel edges of trajectories.

Let $\mathrm{d}_{ij}(t)$ be the Euclidean distance between $i \in \mathcal{X}$ and $j \in \mathcal{X}$ at time t. When $\mathrm{d}_{ij}(t) = \varepsilon$, we say that an *$\varepsilon$-event* occurs. For any ε-event v, we denote by t_v the time when v occurs and $\omega(v)$ the function that returns the two entities that create v. We assume that no two or more ε-events occur at the same time.

Consider an ε-event v; let $\omega(v) = \{i, j\}$. If i and j are further than ε immediately before t_v, then v is a *start ε-event*; if they are further immediately after t_v it is an *end ε-event*. If there is no entity $k \in \mathcal{X}$ located strictly in between i and j at t_v (so $\mathrm{d}_{ik}(t_v) + \mathrm{d}_{jk}(t_v) = \varepsilon$), then we say that v is a *free ε-event*.

▶ **Observation 1.** *The number of ε-events is $O(\tau n^2)$.*

Let G be a group of entities in time interval I that is maximal in size. All entities in G are pairwise ε-connected in the interval I, and hence, there are no free ε-events in G during I. In the arrangement of trajectories from G, we define the height of a face as the length of the longest vertical line segment inside the face. Thus, no face has height greater than ε.

It is also clear that G can begin only at a start ε-event and end only at an end ε-event. Furthermore, we observe that if a start ε-event (or end ε-event) of G is not a free ε-event with respect to the entities in G, then before (or after) the interval I, entities in G are still pairwise ε-connected and we can extend the interval of G. Therefore, G can be a maximal group only if both the start ε-event and end ε-event are free ε-events (but this is not a sufficient condition).

▶ **Observation 2.** *There can be at most one maximal group that starts and ends at a particular pair of start ε-event and end ε-event.*

▶ **Theorem 3.** *For a set $\mathcal{X}$ of n entities, each entity moving along a piecewise-linear trajectory of τ edges, the maximum number of maximal group is $\Theta(\tau n^3)$.*

Proof. Any group G that starts at a start ε-event contains at most n entities. When a free end ε-event involving G occurs, only group G ends but a subgroup of G with fewer entities may continue longer. This can happen at most $n-1$ times. Therefore, the maximum number of maximal groups is $O(\tau n^3)$. Furthermore, there can be $\Omega(\tau n^3)$ maximal groups because the lower bound construction by van Goethem et al. [14] also works for our definition of a group. ◀

The approach to compute all maximal groups is to work on the arrangement $\mathcal{A}$ of line segments that are the trajectories. For a subset $G \subseteq \mathcal{X}$ and interval I, we can remove entities from G that are separated at a face with height larger than ε in I (corresponding to a free ε-event). Only if there are no such faces, the remaining entities in G can be a group. Note that removing entities in G involves removing the corresponding trajectories from the arrangement $\mathcal{A}$, which can cause new faces that are free ε-events.

3 Algorithms for Entities in $\mathbb{R}^1$

In this section, first we consider the case where the trajectories have the same time stamps. We present a basic algorithm that computes all maximal groups in $O(\tau^3 n^6)$ time for entities moving in $\mathbb{R}^1$. Then we present a more efficient algorithm that runs in $O(\tau^2 n^4)$ time. Furthermore, we present an $O(\tau^2 n^4 \alpha(n))$ time algorithm if the vertices of the trajectories have different time stamps.

3.1 Basic Algorithm

We describe a simple algorithm to compute all maximal groups. Let $\mathcal{V}_s$ and $\mathcal{V}_e$ be the sets of all start ε-events and all end ε-events respectively. Fix one event of each type: $\alpha \in \mathcal{V}_s$ and $\beta \in \mathcal{V}_e$. By Observation 2, there is only one maximal group G that starts at α and ends at β. Furthermore, observe that G necessarily contains the entities $\omega(\alpha) = \{a, b\}$ and $\omega(\beta) = \{c, d\}$, and that if G is a maximal group on $I = [t_\alpha, t_\beta]$, then all entities in G are on the same side at time $t_\gamma \in (t_\alpha, t_\beta)$ when a free ε-event γ occurs. We then use the following approach to find G (if it exists):

1. Initialize a set G containing all entities in $\mathcal{X}$.
2. Build an arrangement $\mathcal{A}$ induced by the trajectories of the entities in G on I.
3. A face f in $\mathcal{A}$ contains a free ε-event γ if (and only if) the height of f is more than ε. If f has height larger than ε, test if (the trajectories of) a, b, c, and d, all lie on the same side of f. If not, there is no maximal group G that starts at α and ends at β. If they do pass on the same side, let S denote the set of entities whose trajectories lie on the other side of f. Remove these entities of S from G, and remove their trajectories from $\mathcal{A}$. Observe that new free ε-events may appear because removal of a trajectory from $\mathcal{A}$ merges two faces of $\mathcal{A}$ into a larger one. See Figure 3. Repeat this step until there is no more free ε-event γ with $t_\gamma \in (t_\alpha, t_\beta)$.
4. Check that α and β are now free. If so, G is a maximal group on I, and hence we can report it. If not, G is actually a group during a time interval $I' \supset I$. Hence, G may be maximal in size, but not in duration. We do not report G in this case.

▶ **Theorem 4.** *Given a set $\mathcal{X}$ of n entities in which each entity moves in $\mathbb{R}^1$ along a trajectory of τ edges, all maximal groups can be computed in $O(\tau^3 n^6)$ time using the Basic Algorithm.*

Proof. The number of combination of a pair of start and end ε-events is $O(\tau^2 n^4)$. Building an arrangement from trajectories of entities takes $O(\tau n^2)$ time. Removing a trajectory e

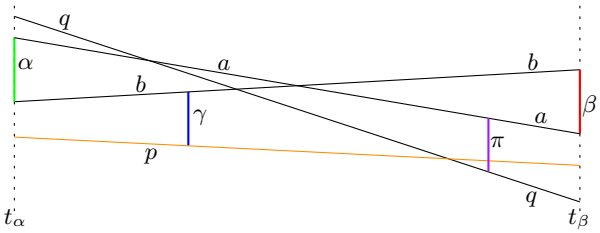

Figure 3 Removing trajectory p (due to the free ε-event γ) causes the ε-event π to become a free ε-event.

and checking new faces in $\mathcal{A}$ takes time proportional to the zone complexity of e: $O(\tau n)$. Since there are at most n trajectories to be removed, the whole process to remove entities for each interval I takes $O(\tau n^2)$ time. Therefore, the running time of the algorithm is $O(\tau^3 n^6)$ time. ◀

3.2 Improved Algorithm

The previous algorithm checks every pair of possible start and end ε-events α and β to potentially find one maximal group. To improve the running time, we fix a start ε-event α and consider the $O(\tau n^2)$ end ε-events β in increasing order. We show that we can check for a maximal group on $[t_\alpha, t_\beta]$ in amortized $O(1)$ time.

We build the arrangement $\mathcal{A}$ for all trajectories, starting from time t_α, and sort the end ε-events β, with $t_\beta > t_\alpha$ on increasing time. We then consider the end ε-events β in this order, while maintaining a maximal set G that is ε-connected in G throughout the time interval $[t_\alpha, t_\beta]$.

Let $\omega(\alpha) = \{a, b\}$ be the entities defining the start ε-event α, and let $G \supseteq \{a, b\}$ be the largest ε-connected set on $[t_\alpha, t_\beta]$. We compute the largest ε-connected set on $[t_\alpha, t_{\beta'}]$ for the next ending event β' as follows. Note that this set will be a subset of G.

Let S be the set of entities that separate from a and b at β. We remove all trajectories from the entities in S from $\mathcal{A}$. As before, this may introduce faces of height larger than ε. For every such face f, we check if a and b still pass f on the same side. If not, there can be no maximal groups that contain a and b, start at t_α, and end after t_β. If a and b lie on the same side of f, we add all entities that lie on the other side of f to S and remove their trajectories from $\mathcal{A}$. We repeat this until all faces in $\mathcal{A}$ that have non-empty intersection with the vertical strip defined by $[t_\alpha, t_{\beta'}]$ have height at most ε (or until we have found a face that splits a and b). It follows that the set $G' = G \setminus S$ is the largest set containing a and b that is ε-connected throughout $[t_\alpha, t_{\beta'}]$. If α and β' are free with respect to G' then we report G' as a maximal group.

Building the arrangement $\mathcal{A}$ takes $O(\tau n^2)$ time, and sorting the ending-events takes $O(\tau n^2 \log(\tau n))$ time. By the Zone Theorem, we can remove each trajectory in $O(\tau n)$ time. Checking the height of the new faces can be done in the same time bound. It follows that the total running time is $O(\tau n^2(\tau n^2 + \tau n^2 \log(\tau n) + R))$ where R is the total time for removing trajectories from the arrangement. Clearly, R is bounded by the complexity of the arrangement: $O(\tau n^2)$. So, the total running time is $O(\tau^2 n^4 \log(\tau n))$.

Further Improvement We can avoid repeated sorting of end ε-events by pre-sorting them in a list, and for each start ε-event, use this list. The list will contain events that do not concern the entities involved in the start ε-event, but this can be tested easily in constant time. Thus, we conclude:

▶ **Theorem 5.** *Given a set $\mathcal{X}$ of n entities in which each entity moves in $\mathbb{R}^1$ along a trajectory of τ edges, all maximal groups can be computed in $O(\tau^2 n^4)$ time.*

Next, we consider finding all maximal groups when the vertices of different trajectories do not have the same time stamps. We use the same idea as in the above algorithm: take one start ε-event α at a time and remove trajectories to find all maximal groups containing $\omega(\alpha)$.

We use a similar strategy to split trajectories vertically into τ cells as in [10], where each cell now contains $O(n)$ segments of trajectories. It follows that the complexity of each cell is bounded by the number of possible intersections between segments: $O(n^2)$. Thus, building the arrangement A still takes $O(\tau n^2)$ time. However, by the Zone Theorem for an arrangement of line segments, removing a trajectory in each cell now takes $O(n\alpha(n))$ time [13], where $\alpha(n)$ is the inverse Ackermann Function. Therefore, the total time to remove trajectories in A is $O(\tau n^2 \alpha(n))$ time and we obtain:

▶ **Theorem 6.** *Given a set $\mathcal{X}$ of n entities in which each entity moves in $\mathbb{R}^1$ along a trajectory of τ edges under the condition that their vertices have different time stamps, all maximal groups can be computed in $O(\tau^2 n^4 \alpha(n))$ time.*

4 Algorithms for Entities in $\mathbb{R}^d$

In $\mathbb{R}^d$ ($d > 1$), it is harder to test whether an ε-event really connects or disconnects because the two entities may be ε-connected through other entities in the group. This observation immediately gives the condition for an ε-event to be *free*. We model our moving entities as a graph where vertices represent entities and an edge exists if two entities are directly ε-connected. As in Parsa [12], we can maintain the graph under edge updates, while allowing same component queries, in $O(\log n)$ time per operation.

To compute maximal groups, we start at a start ε-event α and maintain the connected component $\mathcal{C}$ throughout the sequence of sorted ε-events. At each ε-event β, we remove any vertices that are disconnected from $\mathcal{C}$ and start again from α in case we remove anything. We stop if a and b are disconnected. If α is a free ε-event when we reach β again, we report $\mathcal{C}$ as a maximal group and continue.

We start at $O(\tau n^2)$ ε-events and for each, we process $O(\tau n^2)$ ε-events. We may need to restart this process up to $n-1$ times. In $\mathbb{R}^d$, our approach only examine the ε-events of entities and does not affected by whether the vertices of trajectories have the same time or not, therefore we obtain the same result for both cases:

▶ **Theorem 7.** *Given a set $\mathcal{X}$ of n entities moving in $\mathbb{R}^d$ along a trajectory of τ edges, all maximal groups can be computed in $O(\tau^2 n^5 \log n)$ time.*

5 Algorithms with Linear Dependence on τ

In many real-life situations, the number of vertices in each trajectory is much larger than the number of moving entities. Therefore, the dependence of the algorithm on τ is more important than the dependence on n. Next, we show a simple algorithm that is linear in τ, at the cost of an exponential dependence on n. In particular, our algorithm will compute all maximal groups in $O(\tau n^4 2^n)$ time.

We consider all 2^n subsets of $\mathcal{X}$ in order of decreasing size, while maintaining the set of maximal groups found so far (ordered by increasing starting time). For each subset G we determine the maximal time intervals during which G is ε-connected, and for each such an interval I we check if G is dominated by a maximal group $H \supset G$ on I. If such a set does

not exist, G is a maximal group on I. Notice that we only need to know when the start ε-event and end ε-event of a particular group occured. Therefore, this algorithm applies to both cases where the time stamps of the entities are not the same.

For each subset G, we consider the ε-events generated by the entities in G. We can compute all these $O(\tau n^2)$ ε-events in $O(\tau n^2 \log n)$ time, by sorting the groups of $O(n^2)$ ε-events between two consecutive time stamps separately, and concatenating the resulting τ lists. We then go through the ε-events in order, and check if G is ε-connected at every ε-event. We can easily handle every event in $O(n^2)$ time, by naively checking if the entities in G are ε connected (we can easily improve on this, but the total running time will be dominated by the number of sets anyway). It follows that we can compute the sequence $\mathcal{S}_G$ of maximal time intervals on which G is ε-connected in $O(\tau n^4)$ time. Note that $\mathcal{S}_G$ contains at most $O(\tau)$ such time intervals.

For each interval I in $\mathcal{S}_G$ we now have to check if G is a maximal group during I. The set G is a maximal group on I if and only if there is no maximal group $H \supset G$ on a time interval that contains I. Since we maintain the maximal groups larger than G (and the time interval on which they are a maximal group), ordered by increasing starting time, we can iterate through them once, and extract the maximal groups that are a superset of G. Since, by Theorem 3 there are at most $O(\tau n^3)$ maximal groups, this takes at most $O(\tau n^4)$ time. Let $\mathcal{I}$ denote the set of time-intervals corresponding to those groups, ordered by increasing starting time. We now simply scan through $\mathcal{S}_G$ and $\mathcal{I}$ simultaneously, while maintaining the time interval in $\mathcal{I}$ that started earliest and has not ended yet. For every interval I in $\mathcal{S}_G$ we can then check if G is a maximal group on I in constant time. In total this takes $O(\tau n^3)$ time. Using a similar simultaneous scan we can add the intervals on which G is maximal to our set of maximal groups found so far.

It follows that we can compute all time intervals on which G is maximal in $O(\tau n^4)$ time. Since we do this for all subsets $G \subseteq \mathcal{X}$ we obtain the following result.

▶ **Theorem 8.** *Given a set $\mathcal{X}$ of n entities in which each entity moves in $\mathbb{R}^d$ along a trajectory of τ edges, we can compute all maximal groups in $O(\tau n^4 2^n)$ time, using $O(\tau n^3)$ space.*

6 A Lower Bound on the Maximum Number of Maximal Groups at some Time t

The result in the previous section shows that, when τ is large but n is small, we can improve the dependence on τ from quadratic to linear. However, we pay for this by having an exponential dependence on n. This naturally raises the question whether an algorithm with linear dependence on τ, but polynomial dependence on n, is possible. While we do not know the answer to this question, we present a construction which may indicate that such a result is hard to obtain, if possible at all.

We show that the number of maximal groups that contain a given time t can be exponential in n, provided that τ is sufficiently large. Without the requirement that the maximal groups must span a single moment in time, it is easy to make a construction of trajectories that has a number of maximal groups that is linear in τ, even with just two entities, so it is unbounded in n. Similarly, we can easily construct trajectories that give rise to $2^n - n - 1$ maximal groups (with a group size of at least $m = 2$) that are different in composition using roughly 2^n time stamps by making these groups consecutive. The construction that we present, where many different maximal groups occur simultaneously, is more involved, and shows that there may be $\Omega(\sqrt{2}^n)$ maximal groups simultaneously when there are $\Omega(\sqrt{2}^n)$ time stamps. While the result does not imply any lower bound for the problem of computing all maximal groups,

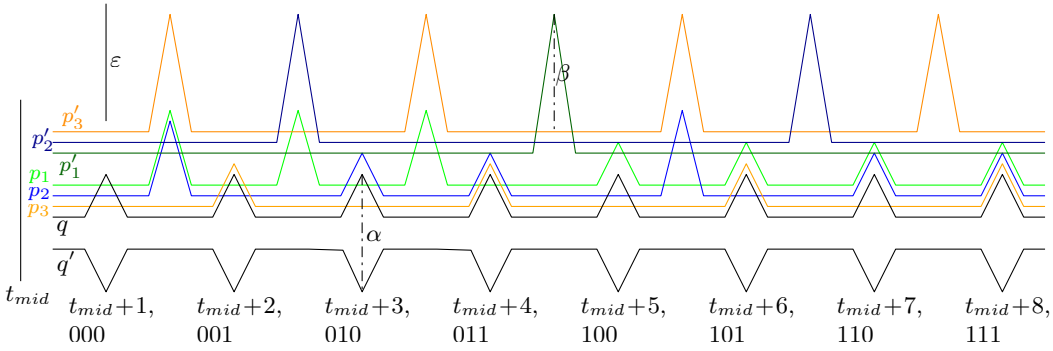

Figure 4 Right half of the lower bound construction for $k = 3$. The times very near the start and end ε-events of q and q' are shown together with the bitstring of the k-max group that ends there. At t_{mid}, all trajectories are in a single point (the trajectories are not shown near t_{mid}).

it suggests that it may be difficult to obtain an algorithm that is linear in τ and polynomial in n. Several natural approaches to the problem (based on, for instance, divide-and-conquer) appear not to work due to this construction and the result on simultaneous maximal groups.

▶ **Theorem 9.** *There exists a set $\mathcal{X}$ of n entities in $\mathbb{R}^1$ whose trajectories are defined by $\Theta(\sqrt{2}^n)$ time stamps, for which the number of maximal groups at some time t is $\Omega(\sqrt{2}^n)$.*

Proof. We use a set of $n = 2k + 2$ entities, denoted $p_1, \ldots, p_k$, $p'_1, \ldots, p'_k$, and q and q'. We are interested in counting the groups that contain q, q', and for each i, exactly one of p_i and p'_i. We call any such group *k-max* and will show that they are all maximal. A k-max group G is encoded by a length-k bitstring where the i-th bit is 1 if $p_i \in G$ and it is 0 if $p'_i \in G$.

We make a construction with the following properties; the half after t_{mid} is illustrated for $k = 3$ in Figure 4:

1. The trajectory of p_i is the reverse of p'_i, with respect to t_{mid} (that is, mirrored in t_{mid}), and vice versa.
2. A k-max group starts and ends at free ε_q-events of q and q'.
3. A k-max group encoded by bitstring B starts a fraction after time $1 + B$ and ends a fraction before time $t_{mid} + 1 + B$, where B is interpreted as a binary number.
4. There are only $O(1)$ trajectory vertices of each trajectory within one time unit.
5. Each k-max group is maximal.

At t_{mid}, all trajectories pass through a single point to ensure they are continuous when mirroring, and they are pairwise directly ε-connected. It is the moment in time for which $\Omega(\sqrt{2}^n)$ maximal groups exist, as we will show. After t_{mid}, the entities q and q' will have 2^k pairs of ε-events: an end ε-event directly followed by a start ε-event. We call these events ε_q-events. Whether these ε_q-events are free for a k-max group G depends on the time and the bitstring, or equivalently, which entities from $p_1, \ldots, p_k$ are in G.

The ε_q-event at a time t is free for a k-max group G if and only if the bitstring corresponding to time t is the same as the bitstring of G. Hence, (assuming that no earlier ε-event ends G) G will end at the time of its bitstring, so a fraction before $t_{mid} + 1 + B$. By symmetry of p_i and p'_i, G will start a fraction after time $1 + B$. In Figure 4, for example, at time $t_{mid} + 4$, the k-max group $\{p'_1, p_2, p_3, q, q'\}$ ends.

The other ε-events of the trajectories are between two consecutive ε_q-events. These ε-events involve the entities of $p'_1, \ldots, p'_k$ and the trajectories need three vertices between ε_q-events. Their presence ensures that only one of p_i or p'_i is in a particular k-max group.

Notice that these ε-events will also be the start or end ε-events of maximal groups that are supersets of a k-max group.

Suppose p'_i is an entity that creates a free ε-event α just before a k-max group G containing p_i ends at $t_{mid} + B$. Obviously, p'_i only needs to create such a free ε-event once and it follows this is only necessary if the previous k-max group G' that ends at $t_{mid} + B - 1$ is not containing p_i. However, other k-max groups that will end after G might contain p'_i. Therefore, to prevent this ε-event becomes free in the duration of those k-max groups, we make entities of $p_1, \ldots, p_k$ that are not in G to keep p'_i ε-connected to all other entities. Still, not all of them are needed to prevent α to become free, but only for each entity p'_h where $h < i$, because by the ordering of the bitstrings, k-max groups contain p'_i and those entities might end after α. See Figure 4: before ε-event α, only p_1 prevents p'_2 from creating a free ε-event (but not p_3). Two k-max groups contain p_1,p'_2 and one of p_3 or p'_3 end after α while k-max group of $\{p'_1, p'_2, p_3\}$ ends before α.

▶ **Claim 10.** *If a maximal group containing time t_{mid} contains at least p_i or p'_i for all indices i, and both p_i and p'_i for at least one index i, then its time interval cannot contain both time h and $t_{mid} + h$ for any integer h.*

Proof. Suppose for contradiction G is a maximal group which contains both p_i and p'_i, and its time interval fully contains an interval $[h, t_{mid} + h]$ for some integer h; suppose further that i is the smallest index for which this is the case. Let B be the bitstring that encodes the entities with indices $1 \ldots i-1$; let $B^- = B0111\ldots1$ be obtained from B by appending a single 0 and $k - i$ 1s and let $B^+ = B1000\ldots0$ be obtained from B by appending a single 1 and $k - i$ 0s. Then G starts not earlier than some time between B^- and B^+, and ends not later than some time between $t_{mid} + B^-$ and $t_{mid} + B^+$. Refer to Figure 4. Hence, there is no integer h such that both h and $t_{mid} + h$ are contained in the time interval of G. ◀

The claim directly implies that all k-max groups are maximal, because by Property 3 they start and end at some time h and $t_{mid} + h$, but adding any other trajectories will cause both p_i and p'_i to be in the group for some i.

Moreover, the ε-events created by entities of $p'_1, \ldots, p'_k$ are also the end ε-events of the $2^k - 1$ groups that have more entities than a k-max group. Let the maximal group contains all entities end at free ε-event β at time $t_\beta = t_{mid} + 2^{k-1} + T$ $(0 < T < 1)$ created by p'_i. By the simmetry of the construction and the ordering of the bitstrings, two groups of $n - 1$ entities not containing either p'_i or p_i will end at time $t_\beta - 2^{k-2}$ and $t_\beta + 2^{k-2}$, respectively. Then, continuing the same process with the two groups recursively will results on other maximal groups with different entities. Since the start and end ε-events of these groups are always start later or end earlier than k-max groups, then these groups are maximal because their interval will not contain interval of other maximal groups. Clearly, the number of these maximal groups is fewer than k-max groups because their ε-events only occur between two consecutive ε_q events. In Figure 4, p'_1 defines β, the end ε-event for a maximal group containing all entities. Then, maximal group that are not contain p_1 or p'_1 will end before or after β, respectively.

To build the construction, all trajectories must have a constant times 2^k vertices for the ε-events of q and q' and a constant number of vertices in between those ε-events. Each trajectory in the construction has $\Theta(\sqrt{2}^n)$ vertices. We conclude that the number of maximal groups that contain time t_{mid} in this construction is at least $2^k = 2^{n/2-1} = \Omega(\sqrt{2}^n)$. ◀

7 Conclusions and Future Work

In this paper we introduced a variation on the grouping structure definition [2] and argued that it corresponds better to human intuition. The number of maximal groups that can arise in a set of n moving entities is $\Theta(\tau n^3)$ in the worst case. We have given an algorithm for trajectories moving in $\mathbb{R}^1$ that computes all maximal groups and runs in $O(\tau^2 n^4)$ time. In $\mathbb{R}^d$, our algorithm runs in $O(\tau^2 n^5 \log n)$ time. For the more general case where the input trajectories do not have time-aligned vertices, the algorithm for trajectories in $\mathbb{R}^1$ can be extended at the cost of an extra factor of $\alpha(n)$, while the same result still holds for trajectories in $\mathbb{R}^d$.

Furthermore, we presented an algorithm that has only linear dependence in τ, at the expense of exponential dependence in n. Since collections of trajectories are often very large in the number of time stamps and not necessarily in the number of trajectories, this algorithm or a practical variation on it may still be useful. This algorithm is not affected by whether or not the vertices of the trajectories are aligned in time.

The trade-off in the dependence on n and τ gives rise to interesting open problems. Most importantly, is it possible to develop an algorithm whose running time is linear in τ and polynomial in n? Similarly, can we realize subquadratic dependence on τ without having exponential dependence on n? In general, what trade-offs are possible?

Future work includes implementing our algorithms and experimentally showing the differences between the definition of groups in [2] and our refined definition, both qualitatively and quantitatively. It would also be interesting to develop an output-sensitive algorithm that uses considerably less time if the output is small, or under realistic input assumptions. Finally, it would be interesting to investigate whether one can develop algorithms that take geodesic distance into account to define direct ε-connectedness instead of the straight-line distance, as was done for the previous definition of a group [10].

References

1 Marc Benkert, Joachim Gudmundsson, Florian Hübner, and Thomas Wolle. Reporting flock patterns. *Computational Geometry*, 41(3):111–125, 2008.

2 Kevin Buchin, Maike Buchin, Marc van Kreveld, Bettina Speckmann, and Frank Staals. Trajectory grouping structure. *Journal of Computational Geometry*, 6(1):75–98, 2015.

3 Joachim Gudmundsson, Patrick Laube, and Thomas Wolle. Computational movement analysis. In *Handbook of Geographic Information*, pages 423–438. Springer, 2012.

4 Joachim Gudmundsson and Marc van Kreveld. Computing longest duration flocks in trajectory data. In *Proc. 14th ACM International Symposium on Advances in Geographic Information Systems*, GIS'06, pages 35–42, 2006.

5 Joachim Gudmundsson, Marc van Kreveld, and Bettina Speckmann. Efficient detection of patterns in 2D trajectories of moving points. *GeoInformatica*, 11:195–215, 2007.

6 Yan Huang, Cai Chen, and Pinliang Dong. Modeling herds and their evolvements from trajectory data. In *Geographic Information Science*, volume 5266 of *LNCS*, pages 90–105. Springer, 2008.

7 San Hwang, Ying Liu, Jeng Chiu, and Ee Lim. Mining mobile group patterns: A trajectory-based approach. In *Advances in Knowledge Discovery and Data Mining*, volume 3518 of *LNCS*, pages 145–146. Springer, 2005.

8 Hoyoung Jeung, Man Yiu, Xiaofang Zhou, Christian Jensen, and Heng Shen. Discovery of convoys in trajectory databases. *PVLDB*, 1:1068–1080, 2008.

9 Panos Kalnis, Nikos Mamoulis, and Spiridon Bakiras. On discovering moving clusters in spatio-temporal data. In *Advances in Spatial and Temporal Databases*, volume 3633 of *LNCS*, pages 364–381. Springer, 2005.

10 Irina Kostitsyna, Marc van Kreveld, Maarten Löffler, Bettina Speckmann, and Frank Staals. Trajectory grouping structure under geodesic distance. In *Proc. 31st International Symposium on Computational Geometry, SoCG 2015*, pages 674–688, 2015.

11 Zhenhui Li, Bolin Ding, Jiawei Han, and Roland Kays. Swarm: Mining relaxed temporal moving object clusters. *PVLDB*, 3(1):723–734, 2010.

12 Salman Parsa. A deterministic $O(m \log m)$ time algorithm for the Reeb graph. In *Proc. 28th Annual Symposium on Computational Geometry*, SoCG'12, pages 269–276, 2012.

13 Micha Sharir and Pankaj K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications*. Cambridge University Press, 1995.

14 Arthur van Goethem, Marc van Kreveld, Maarten Löffler, Bettina Speckmann, and Frank Staals. Grouping time-varying data for interactive exploration. In *Proc. 32th Annual Symposium on Computational Geometry*, SoCG'16, pages 61:1–61:16, 2016.

15 Yu Zheng and Xiaofang Zhou. *Computing with Spatial Trajectories*. Springer, 2011.