Group diagrams for representing trajectories

Maike Buchin^a, Bernhard Kilgus^a and Andrea Kölzsch^{b,c}

^aDepartment of Mathematics, Ruhr University Bochum, Bochum, Germany; ^bMax Planck Institute for Ornithology, Radolfzell, Germany; ^cDepartment of Biology, University of Konstanz, Konstanz, Germany

ABSTRACT

Given the trajectories of one or several moving groups, we propose a new framework, the *group diagram (GD)* for representing these. Specifically, we seek a minimal GD as a concise representation of the groups maintaining the spatio-temporal structure of the groups' movement. A GD is specified by three input values, namely a distance threshold, a similarity measure and a minimality criterion. For several variants of the GD, we give a comprehensive analysis of their computational complexity and present efficient approximation algorithms for their computation. Furthermore, we experimentally evaluate our algorithms on GPS data of migrating geese. Applying the proposed methods on these data sets reveals how the GD concisely represents the movement of the groups. This representation can be used for further analysis and for the formulation of new hypotheses for further ecological research, such as differences in movement patterns of groups on different surfaces or the shift of migration routes over several years. We use different similarity measures to summarize the migration routes of (i) a goose family for one migration period and to summarize (ii) the migration routes of one individual for several migration periods or (iii) the migration routes of several independent individuals for one migration period.

KEYWORDS

Movement analysis;
trajectory analysis;
computational geometry;
Fréchet distance; equal-time
distance

1. Introduction

The amount of available movement data such as GPS data collected by mobile devices has increased massively over the last years. Movement ecology, for instance, is a growing field at the edge of biology and GIScience which benefits from new possibilities of tagging animals with smart devices collecting location data and several other parameters. Analyzing this data poses new algorithmic challenges (Gudmundsson *et al.* 2012). One of the challenges is to (visually) represent the large amount of data, such that the representation is as concise as possible, but still preserves the information needed for further analysis and processing. In particular, this challenge arises for movement trajectories of several interacting entities.

In this paper, we take on this challenge and present a concise representation for data of groups of moving objects. Our approach is the following: We assume we are given the trajectories of several moving objects, called *entities*. Each trajectory consists of a series of sampled locations. By linearly interpolating in between each two consecutive timestamps,

CONTACT Bernhard Kilgus bernhard.kilgus@rub.de

the corresponding trajectory becomes a polygonal line. We assume we are given k trajectories, each of complexity n (number of vertices) forming one or several (overlapping, i.e. splitting and merging) groups. We introduce a new framework named the *group diagram* (GD) as a means of a concise representation of these groups. In particular, the GD allows for a concise visual representation.

The group diagram identifies groups moving together that can be represented by one of the individuals. Its definition is quite general and in particular allows us to consider different ways of processing time. More specifically, we define the GD as a geometric graph (see the next section for precise definitions) that represents all of the input trajectories in the sense that for each input trajectory there is a similar path in the GD. Furthermore, the GD should have a minimal size with respect to the number/length of the edges in the geometric graph.

1.1. Related work

Two related notions to the GD are the grouping structure and flow diagrams. The grouping structure is defined as the unique graph which represents all density-connected groups traveling at equal-time (Buchin *et al.* 2015). It can be seen as a specialization of the GD that uses density connectedness at equal timestamps as inner group distance. As such the grouping structure is well suited for grouping entities moving together, but not for entities traveling, e.g. commuting or migrating, at different times. Also, in some settings, such as commuting, pairwise distance is more suitable than density connectedness. For instance, commuters moving in slow traffic along a major road would form a group in the grouping structure, even though individual commuters in this group may be far away from each other. Also note that density connectedness and equal-time alignment of the time-stamped locations are inherent to the grouping structure (which is based on the Reeb graph). Hence, it would not generalize to other distance measures, as the GD does. Moreover, the Reeb graph that forms the grouping structure does not consist of representatives that capture the movement of each entity because density connectedness does not allow to represent a group by one individual that is close to all in the group. In contrast, for each entity, the group diagram contains a path which is similar to the corresponding trajectory of the entity.

Buchin *et al.* (2015) gave efficient algorithms for computing the grouping structure, analyzed its complexity, and demonstrated its usefulness on data sets of deer, elk, and cattle. The grouping structure was later generalized by Kostitsyna *et al.* (2015) for the geodesic distance and by van Goethem *et al.* (2016) for varying parameters. Furthermore, van Kreveld *et al.* (2016) refined the grouping structure by restricting the density connectedness to within a group. They claim that this is a more intuitive definition since Buchin *et al.* (2015) allow two entities of one group to be density connected via other entities not in the group. However, a grouping structure based on the refined definition is less efficient to compute. Wiratma *et al.* (2018) experimentally compared the original and the refined version and showed that the magnitude of change between the original and refined grouping structure increases if the density in the crowd increases. In order to detect trends in one-dimensional time-varying data, van Goethem *et al.* (2017) built on the work of Buchin *et al.* (2015) and van Kreveld *et al.* (2016).

A flow diagram is a minimal diagram (in the number of vertices) representing segmentations of all input trajectories. In a flow diagram, vertices represent segmentation criteria and

edges transition between criteria (Buchin *et al.* 2016). As a simple example, a trajectory of a person in one day could be roughly segmented to *home-work-home*. The flow diagram can be seen as a generalization of the GD (after switching between vertices and edges) where criteria are more general than the distance between trajectories. However, for this, one would need to consider an infinite number of criteria of the form ‘traveled this route at this time’, which is not computationally feasible. Buchin *et al.* (2016) showed that deciding if a flow diagram of a certain size exists is NP-hard (even W[1]-hard in the number of trajectories). Hence, they give efficient heuristics for computing flow diagrams and evaluate these on a data set of football players.

Further, related approaches for detecting and representing groups of moving objects are flocks (Gudmundsson and van Kreveld 2006, Gudmundsson *et al.* 2007, Benkert *et al.* 2008, Vieira *et al.* 2009), convoys (Jeung *et al.* 2008), swarms (Li *et al.* 2010) and herds, (Huang *et al.* 2008, Baltzer *et al.* 2008). They differently incorporate time and space in their definition of a group. For instance, some concepts require that the entities stay in a disc whereas others use density connectedness (Ester *et al.* 1996) as grouping criterion.

The only concept that captures splitting and merging of (sub)groups is the herd concept (Huang *et al.* 2008, Baltzer *et al.* 2008). In contrast to our approach, the authors use density-connected groups and define splitting and merging of groups based on a definition of recall and precision measure of a herd. Furthermore, this concept only exhibits the evolution of a group for discrete timestamps whereas the GD graphs show the evolution of one or several groups continuously from the beginning to the end of the movement of each entity. Moreover, no representatives which express the movement of each entity are computed, whereas the GD framework does compute such representatives. Baltzer *et al.* (2008) present OLAP (online analytical processing) for trajectories to detect groups based on geographic overlap and intersection. Their approach is based on mapping the trajectories on a two-dimensional spatial grid. In contrast, our approach works with precise geometric distance and allows for different ways of handling the temporal component.

Computing a GD using the Fréchet distance is also related to map construction algorithms wherein the goal is to determine the underlying network of a set of trajectories (Ahmed *et al.* 2015).

Similar modeling choices, i.e. the choice of the similarity measure, the minimality condition, and what constitutes edges also occur in the problem of finding a representative of a set of similar trajectories. Approaches to this problem are, for instance, the median trajectory (Buchin *et al.* 2010), the middle trajectory (Ahn *et al.* 2016), or the central trajectory (van Kreveld *et al.* 2015).

1.2. Contribution

We present a new framework called group diagram (GD) to concisely represent the movement of one or several moving groups. As outlined above, our framework inherently differs from all previous methods and approaches of representing the movement of groups. In contrast to the grouping structure, it can be applied to movement data of entities moving at different periods of times and it captures the pairwise distance between the entities, whereas the grouping structure is based on density connectedness. Formally, the flow diagram allows for similar computations, but it cannot be applied in

practice to compute all occurrences of splits and merges within a group as one needs to consider an infinite number of criteria. Furthermore, in contrast to all approaches discussed above, our framework computes geometric representatives such that each individual movement is expressed by at least one of these representatives.

Our work is motivated by the problem of finding such a representation for the movement of groups of migrating greater white-fronted geese (*anser a. albifrons*). For this specific research question, we present experimental results for one goose family and for groups of independent individual geese. The input data consists of one trajectory for each goose, given by a number of time-stamped GPS-locations. In order to obtain a concise representation, we compute a minimal GD for the input trajectories, and for a visual representation, we plot the representatives of the GD on a map. This representation, for instance, shows typical migration routes for this species of geese or highlights where the goose family was flying closely together and where they were moving with larger pairwise distance, i.e. more separated. We vary the distance threshold of the GD to investigate for which distance the family stayed 'together' during the whole migration route or for a given percentage of the whole observation period. Furthermore, we generated synthetic data similar to the input to examine the scalability of the methods and their implementation, as real data sets with more entities are not available.

Our framework allows us to incorporate several problem sensitive specifications such as the criterion specifying the conciseness of the group diagram and the similarity measure between the moving objects. Therefore, we believe the framework can be applied to a wide range of research questions in the field of movement analysis and beyond.

In this paper, we present the general framework and algorithmic idea to compute the GD. Subsequent to the analysis of the computational complexity, we introduce specific algorithms for three similarity measures. These measures are the equal-time distance, the similar-time distance, and the Fréchet distance. The motivation behind focusing on these measures is that they allow to compute a GD for different types of groups. These types are groups where all objects move simultaneously (equal-time distance), within a given time-shift (similar-time distance) or time-independent, that is at different periods of time (Fréchet distance). As minimality criteria, we consider the minimal total edge length of the GD and the minimal number of edges.

The definition of the framework and the general algorithmic approach is described in Section 2. In Section 3, we show that computing a minimal GD is NP-hard for all the variants considered in detail. Hence, we develop approximation algorithms in Section 4 and Section 5. The results of applying the GD on data of migrating geese are presented in Section 6.

This paper extends the results of (Buchin *et al.* 2018) presented at the International Workshop on Computational Transportation Science in 2018. Here, we give a more detailed presentation of the general algorithmic approach and a comprehensive analysis of the computational complexity of all variants of the GD we consider. Furthermore, we explore the use of the Fréchet distance as a similarity measure for GD. This measure allows us to compute GDs for trajectories recorded within different periods of time as this distance only considers the ordering of the timestamps (vertices) along the trajectory and not the specific time at each location. For instance, this is the case if the input consists of migration routes of one individual recorded during several different migration periods.

Considering one migration period, we can summarize the migration routes of several independent individuals which do not necessarily migrate simultaneously.

2. Definition and algorithmic approach

2.1. Preliminaries

First, we introduce our terminology and then we formally define GD. A *trajectory* describes the movement of an entity over time. This means, it entails a temporal and a spatial component. The movement of an entity is assumed to be continuous, but it is measured at a discrete set of times. Formally, a *trajectory* $T = ((p_1, t_1), (p_2, t_2), \dots, (p_n, t_n))$ of complexity n is an ordered sequence of positions in time t_i and space p_i , whereby the space is two- or three-dimensional. In this paper, space is always two-dimensional. We assume that an entity moves with constant velocity on a straight line between two time-stamped locations. That is, we use linear interpolation between two time-stamped locations to obtain a continuous movement from p_1 to p_n . Therefore, we can interpret T as a polygonal curve. In the following, we denote the time-stamped locations as vertices of T . A *subtrajectory* refers to an arbitrary but connected part of the trajectory starting and ending at arbitrary points along the trajectory. We call a subtrajectory starting and ending at vertices a *restricted* subtrajectory. A subtrajectory that is not necessarily restricted is an *arbitrary* subtrajectory. A *segment* refers to a subtrajectory consisting only of two consecutive vertices of the trajectory and the linear interpolation between them.

A *cluster* is a set of trajectories that are similar (under some similarity measure) to one *representative* of the cluster. A cluster representative is thus a subtrajectory within a cluster to which all other subtrajectories in the cluster have distance at most d . A *cluster curve* is a subtrajectory within a cluster. Given a cluster representative τ , we denote by $c(\tau)$ the set of all cluster curves of the corresponding cluster. For a set of representatives A , $c(A)$ is the union of all cluster curves represented by a representative in A .

2.2. Definition of the group diagram framework

We propose the following general definition for a group diagram:

Definition 2.1. A group diagram (GD) is a geometric graph with vertices augmented by a temporal component, which represents all input trajectories $\mathcal{T}$. We say the graph represents a trajectory $T \in \mathcal{T}$ if there is a similar path P in the graph, that is T and (the geometric representation of) P are similar under a given similarity measure. A GD is minimal if it is minimal in size, either with respect to its number of edges or the total length of edges.

Note that a group according to Definition 2.1 can consist of not only several individuals, but also one individual and its movement for several different periods of time. Furthermore, the definition allows to make several choices: how to represent vertices and edges of the GD, the similarity measure, and whether to minimize the number or the total (geographical) length of the edges.

We consider GDs that are built from the input trajectories, i.e. edges of the GDs are represented by subtrajectories of the input. These give realistic representatives in the sense that they show an actual movement of an entity, despite interpolation between recorded locations and measurement errors. In contrast, newly generated representatives do not satisfy natural restrictions in general and might, for instance, pass through a lake although this is not possible for the specific entities. Furthermore, two edges share a vertex if the endpoints of the corresponding subtrajectories are within distance d from each other. Endpoints of edges without a d -distance neighbor have degree one. Vertices in the graph are thus embedded as the set of endpoints of incident edges. We will use such graphs in the following. Note that we could transform these into planar embedded graphs, for instance by connecting all points of a point set S of a vertex to the midpoint of S .

As similarity measure, we consider three popular measures on trajectories: the Fréchet distance, equal-, and similar-time distance. All of these measure the similarity as the maximum distance under certain alignments of the temporal component: equal-time aligns equal timestamps, similar-time allows a bounded shift in time, and the Fréchet distance allows for arbitrary shifts, however, still respecting the ordering in time (see Section 4 for a formal definition). Figure 1 shows GDs for the three distance measures for an input of four trajectories.

The generality of our definition allows to apply it in different settings, e.g. to entities moving at the same or at different times. Also note that the information on the number of subtrajectories similar to one edge of the GD (*representation strength* of an edge) can be used, for instance, to indicate the most commonly traveled routes by adjusting the thickness of each edge according to its representation strength when drawing the GD.

Minimizing the number of edges or their total length seems intuitively reasonable, especially if we want to obtain, for instance, a concise representation of the group movement plotted on a map. However, both can lead to unwanted effects: When minimizing the number of edges, similar parts of the movement of two entities might not be captured and when minimizing the total length, two unconnected vertices within distance d represent the same part of the input as one segment of length d . For instance, the shortest representation of a single trajectory would consist of a series of points at distance $2d$ (see Figure 2 for an illustration). Thus, we need to formulate further requirements to prevent these effects. The requirements are the following:

- When minimizing the number of edges, we require the minimality criterion to be fulfilled for every local part of the input: Given a GD $\mathcal{G}$, an arbitrary subtrajectory τ of one of the representatives in $\mathcal{G}$, and the corresponding cluster $c(\tau)$, we use $G_\tau^* := c(c(\tau))$ to denote the union of all clusters represented by a curve in $c(\tau)$. We intersect G_τ^* with the GD $\mathcal{G}$ to obtain all representatives $A_\tau := \mathcal{G} \cap G_\tau^*$. The *local minimality criterion* now requires that for each τ as described above the resulting set A_τ is a minimal representation for $c(A_\tau)$.
- When minimizing length, we require that no clusters are artificially split up to reduce the length. Formally, we require that no subgraph of the GD can be replaced by a subgraph with less connected components but possibly larger total edge length.

In the following, we consider the problem with these additional requirements.

Figure 1. GDs for an input of four trajectories. As depicted in the upper figure, the upper two trajectories show a movement which starts at 00:00, the third trajectory shows a movement which starts with a delay of 10 min and the fourth trajectory shows a movement with a delay of 100 min in comparison to the first two trajectories. Note that the period of time between two consecutive vertices is 10 min. The GDs for the three different similarity measures are marked in red. For the equal-time distance, the upper two trajectories are summarized if they are within a spatial distance of at most d . A more rigorous summary of the trajectories is not possible, since the timestamps of the other trajectories do not align with the timestamps of the upper two. When a similar-time distance with an allowed time shift of 10 min is applied, it is possible to summarize the upper three trajectories by one representative whenever the spatial distance of the trajectories is at most d . Finally, when applying the Fréchet distance, only the spatial distance is considered, independent of the specific time of the movements.

Figure 2. Counterintuitive GD with a minimal number of edges (left) and minimal edge length (right).

2.3. General algorithmic approach

First, we observe that each edge in a GD corresponds to a representative of a cluster of subtrajectories from the input. This observation leads to the following general algorithmic idea: We subdivide the problem of computing a minimal GD into a *clustering* and a *covering* problem. That is, we first detect a set of cluster representatives and then select a minimal set of these clusters whereby the union covers the complete input. Thus, our

Figure 3. The workflow of the general algorithmic approach to compute a minimal GD. All processed data is marked with trapezoids and the computation steps with rectangles. The input data is processed first to obtain a sufficiently fine-grained segmentation. Then, the clustering step is applied to this augmented data and outputs a *Set-Cover* instance which is solved in the last step to obtain a minimal GD.

approach is to construct and solve a *SET-COVER* instance with universe $\mathcal{U}$ consisting of all segments of the input trajectories.

To make this approach computationally feasible, we need to ensure that all representatives which are part of an optimal solution start and end at some vertex. However, in general, this does not hold for the segmentation of the trajectories given by the input vertices. Here, a representative which is part of a minimal GD might start and end in between two input vertices along the trajectory. We, therefore, need to preprocess the input data to obtain a sufficiently fine-grained segmentation. We call the obtained higher resolution data the *augmented input*.

To take the two different minimality criteria into account, we can formulate a weighted *SET-COVER* problem in which we assign a weight to each representative (subset) depending on which minimality condition we choose: unit weight for the number of edges, and length of representative for edge length. The workflow of this general algorithmic approach is summarized in Figure 3.

Note that the criterion for minimizing the edge number and the criterion for minimizing the total edge length is contrasting. For a minimal length, we require that the representatives are not cut too often whereas to fulfill the condition of minimal edge number, we have to force some cuts so that we can ensure a minimal representation also for an arbitrarily small part of the input data. In Section 4.1 we formally define a *relevant* cluster representative and show how these representatives optimally balance the number of cuts, such that a solution which consists of relevant cluster representatives fulfills the additional criteria for both minimization goals.

Solving Set-Cover As the *SET-COVER* problem is *NP*-complete, there is no exact polynomial time algorithm to solve the problem unless $P = NP$. We use an implementation of a greedy algorithm to approximate the constructed *SET-COVER* instance. A greedy algorithm takes the largest subset in each step and deletes the points of the universe already covered from the other subsets which are not selected yet. Suppose m is the minimum number of subsets (or the minimum total weight). Then, the greedy algorithm returns a solution of size less than $m \log(u)$, where u is the cardinality of the universe. In fact, *SET-COVER* cannot be efficiently approximated better than to a factor of $(1 - o(1)) \log(u)$ unless $P = NP$. The runtime of the greedy algorithm is bounded by $\mathcal{O}(su \min\{s, u\})$, where s is the number of the given subsets.

3. Computational complexity

In this section, we prove that computing the minimal GD is *NP*-hard for all variants we consider. For this, we reduce an *NP*-complete variant of the Dominating-Set problem to

the decision version of our problem. The decision version is formulated in the following definition. This section justifies developing approximate solutions in the following and can be skipped by the reader not interested in computational complexity theory.

Definition 3.1. Given k trajectories and an integer value l , the Group-Diagram problem is to decide whether there is a collection of subtrajectories S of total size l , such that every input trajectory has distance at most d to at least one path in the group diagram consisting of the subtrajectories in S . Again, if an endpoint of a subtrajectory and a start point of another subtrajectory are within distance at most d they share a vertex in the group diagram.

In this definition, the size of a GD is either the smallest integer greater than or equal to the total length of the edges, which we denote by GD-Min-Length or the number of edges in the GD, which we denote as GD-Min-Edgenumber. The distance of a trajectory to a path in the GD refers either to the Fréchet distance or to equal-time distance.

For the reduction to prove NP-hardness, we need to recall some definitions and results from the literature. We define the Dominating-Set problem and a variant with more geometric structure. This variant is still NP-complete, and thus, we can reduce from this problem and exploit the geometric structure to show NP-hardness for GD-Min-Length and GD-Min-Edgenumber for a variant where all trajectories are parallel. Hence, we can conclude that both GD-Min-Length and GD-Min-Edgenumber are NP-hard to decide (and to compute) for arbitrary input trajectories.

Definition 3.2. Given a graph $G = (V, E)$. A dominating set D is a subset of V such that every vertex from V not in D is adjacent to at least one vertex in D . Given an integer l , the DOMINATING-SET problem is to decide whether there is a dominating set of size l .

DOMINATING-SET is known to be NP-complete for general graphs. However, the problem remains NP-hard for grid graphs, which was proven by Clark et al. (1990):

Definition 3.3. Let $\mathcal{D} = \{(x, y) \in \mathbb{R}^2 \mid x \text{ and } y \text{ are multiples of } d\}$, where $d \in \mathbb{R}$. A grid graph is a graph $G = (V, E)$, $V \subset \mathcal{D}$ and wherein there is an edge between two vertices if and only if they are within distance d .

Lemma 3.4. DOMINATING-SET for grid graphs is NP-complete.

With these preliminary remarks, we can prove that Group-Diagram is NP-hard by a reduction from DOMINATING-SET for grid graphs.

Theorem 3.5. Given an integer l , deciding whether there exists a GD of size l is NP-hard for both l denoting the edge length and l denoting the edge number, and for both Fréchet distance and equal-time distance as similarity criteria.

Proof. Fréchet distance: Given an arbitrary grid graph $G = (V, E)$ with $V = \{v_1, v_2, \dots, v_k\}$, we construct an instance of Group-Diagram as follows: For each $v \in V$ we place a horizontal segment with length less than d in the plane wherein the coordinates of

Figure 4. A grid graph and its corresponding placement of segments (right). The solution of the DOMINATING SET and the GROUP DIAGRAM instances is highlighted in red.

the starting point equal the coordinates of v . See Figure 4 for an example. On the one hand, if two vertices in G are adjacent, then the corresponding segments are at Fréchet distance at most d as the segments are equally long and parallel. On the other hand, if the Fréchet distance between two (sub)segments, s_1 and s_2 , is at most d , then the vertices in G , that correspond to s_1 and s_2 , are adjacent in G . This follows from the construction of the segments. If two vertices v_1, v_2 are not adjacent in G , then any point along the corresponding segment to v_1 has distance greater than d to any point along the segment corresponding to the vertex v_2 .

Now suppose there is a dominating set $I = \{v_{i_1}, \dots, v_{i_l}\}$ of size l for G . Let $\hat{I} = \{s_{i_1}, \dots, s_{i_l}\}$ denote the set of the corresponding segments. For each $s \notin \hat{I}$ there is a vertex $v \in I$ adjacent to the corresponding vertex v_s and it follows that s and s_v are within distance d . Therefore, any input trajectory is represented by at least one 'path' consisting of only one input segment. Now consider a set J of (sub-)segments of size l which represents the whole data. Particularly for each starting point p of the constructed segments, there is a (sub-)segment in J with starting point q , where $\text{dist}(p, q) \leq d$. As the starting points of the segments equal the vertices in the graph, a dominating set of size l for G consist of vertices corresponding to the (sub-)segments of J .

Note that a minimal solution in this setting also fulfills the local minimality criterion. This follows from the observation that for each subtrajectory τ_s of a given trajectory T , the cluster $c(\tau_s)$ contains the same (shrunk) cluster curves as $c(T)$. Therefore, for any representative r of the GD, the part G^* relative to any subtrajectory of r is represented in a minimal way by the corresponding subsegments of the GD.

For proving NP-hardness of Group-Diagram using total edge length as size, we fix the distance $d = 1$ and the length of the segments induced by the vertices of G to $l = d$. To fulfill the additional condition for GD of minimal total edge length, observe that a solution consists of whole segments as the segments are parallel and starting points are within distance $d = 1$. Thus, each representation of two segments within distance d must have a length of at least 1 and a representation of several subsegments can be replaced by a smaller subgraph in the GD by choosing one segment as representative. Hence, a GD which fulfills the demanded is composed of representatives which correspond to a whole segment only. Furthermore, a solution of Group-Diagram with total edge length l consists of exactly l segments where the corresponding vertices are a solution of DOMINATING-SET of size l in G . Obviously, a solution of Dominating-Set of size l implies a solution of Group-Diagram of total edge length l , by choosing the corresponding segments as GD edges.

This proves NP-hardness of the problem of finding a minimal group diagram for a given set of input trajectories. Note that NP-hardness is shown for the special case of parallel, equal size trajectories with only one segment (and in case of minimal edge length with distance 1).

Equal-time distance: To prove the NP-hardness, note that a GD with a limited number of segments, in particular, provides us with a representation of the objects for every timestamp with limited size. Thus, it suffices to prove the NP-hardness of the problem of deciding if there is a representation of a given size for a single timestamp. This again can be easily proven by a reduction from Dominating Set (see Definition 3.2) by identifying the location of the entities for a single timestamp with the vertices of a graph G where we insert an edge between two vertices if and only if the corresponding entities are within distance at most d . Evidently, a representation of the objects of size l equals a dominating set in G of size l . $\square$

4. Group diagrams based on the Fréchet distance

Following the idea presented in Section 2.3, we construct and then approximately solve a SET-COVER instance. Therefore, we obtain approximation algorithms in both cases, minimizing the total size and minimizing the length of the GD. When minimizing length for the Fréchet distance, we additionally make a small additive error in the construction for each edge of the GD, see Lemma 4.2.

Recall that the Fréchet distance between two polygonal curves T and S parameterized over $[0, 1]$ is defined as the infimum over all reparameterizations α and β of $[0, 1]$ of the maximum over all $t \in [0, 1]$ of $\|T(\alpha(t)) - S(\beta(t))\|$ (Alt and Godau 1995).

Typically, the Fréchet distance is illustrated by a man walking his dog. The Fréchet distance equals the shortest length of a leash that allows the man and the dog to walk on their curves from beginning to end. Both the man and the dog are only allowed to move forward along their curve or to stop at some point of the curve and continue moving forward. The Fréchet distance between two polygonal curves of complexity n can be computed in $O(n^2 \log n)$ time (Alt and Godau 1995).

To compute a minimal GD with Fréchet distance as similarity criterion, we use a sweep algorithm with moving points a_T and b_T along each trajectory T and report all *relevant* clusters represented by the subtrajectory between the current positions of a_T and b_T as described in (Buchin *et al.* 2011). In the following, we formally define a relevant cluster representative. Intuitively, a representative of a cluster is relevant if any extension of the representative causes a change of the composition of the trajectories that are part of the cluster. A sufficiently fine-grained segmentation of the input trajectories, such that all relevant representatives start and end at a vertex, is obtained by propagating each vertex of the input trajectories to all other trajectories within d -distance to this vertex and by computing all d -distance points between segments of the input. If the reader is more interested in the overall idea and the applications of the concept, Section 4.1 can be skipped.

4.1. Segmentation

As proposed in Section 2.3, we need to insert new vertices along the trajectories to obtain a minimal representation where all representatives start and end at a vertex. See Figure 5 for an illustrating example of an insufficiently fine-grained input.

Before we develop a strategy of inserting vertices, we precisely define a relevant cluster representative. We will show (Lemma 4.2) that it suffices to consider only these representatives. Recall that a cluster representative is a subtrajectory of the input and cluster curves are the subtrajectories within a fixed distance d to the representative. We use d -tube $\mathcal{T}_d(\tau)$ to refer to the d -neighborhood of a subtrajectory τ , that is the set of all points with distance $\leq d$ to a point on τ .

Definition 4.1. A cluster representative τ which represents the cluster $c(\tau)$ is irrelevant, if it can be extended to τ' such that $c(\tau')$ contains only extended curves of $c(\tau)$ and $|c(\tau)| = |c(\tau')|$ and such that no other trajectory T with $T \cup \mathcal{T}_d(\tau) = \emptyset$ contains a subtrajectory inside one of the d -tubes around the cluster curves of $c(\tau')$. If a cluster representative cannot be extended in such a way, we say the representative and the corresponding cluster are relevant.

Figure 6 shows some examples of relevant and irrelevant clusters.

Lemma 4.2. When minimizing size, there always exists a minimal GD solution where edges correspond to relevant cluster representatives. When minimizing length, this solution adds at most an additive error of $2de$, where d is the distance threshold and e is the number of edges of an optimal solution.

Figure 5. Input vertices are not sufficient for a minimal representation. No representative starting and ending at the input vertices represents all three subtrajectories in the middle part as the endpoints of the segments are within distance greater than d .

Figure 6. Examples of relevant and irrelevant clusters. The cluster marked in red in the left figure can be extended to the cluster marked in green in the middle figure. Therefore, the red cluster is irrelevant. The green clusters in the middle and right figure can not be extended unless one of the curves is dropped from the cluster (middle) or unless the distance of one cluster curve to a subtrajectory, which is not in the cluster falls below a given distance threshold (right).

Proof. For an irrelevant cluster we can always extend the representative such that the cluster is extended as well and none of the cluster curves is represented by another trajectory not in the cluster in a possibly better way with respect to the total size of the GD. In terms of minimal length, it is possible to shrink a relevant cluster representative τ such that it still represents the same cluster $c(\tau)$, but the amount of shrinking is bounded by $2d$ as otherwise, the distance to the cluster curves is greater than d . $\square$

We now describe how to obtain a sufficiently fine grained segmentation such that all relevant clusters start and end at vertices.

Inserting New Vertices We consider two triggers for inserting a new vertex. First, for every vertex v of the input data, we add a vertex to every segment which has a distance to v less than or equal to d (see Figure 7a) at the point along the segment where the distance to v is minimal (type 1). Second, we add a new vertex, if the distance between the two segments is less than d for the first time and if the distance exceeds d again (type 2) (see Figure 7b). Type 1 introduces 0 or 1 new vertex for each vertex-segment pair and type 2 introduces 0, 2 or 4 vertices for each pair of segments.

We first observe that only one step of inserting new vertices does not suffice. A counterexample is shown in Figure 8b. However, two steps of inserting new vertices are sufficient.

Figure 7. Inserting new vertices. The vertices from the input are shown as disks whereas the newly added ones are marked as squares.

Figure 8. The GD consisting of the representatives marked in red does not have minimal edge length or violates the local minimality criterion. This cannot be avoided in this example, if only input vertices (disks) and vertices of the first vertex insertion step (squares) are allowed to bound the representatives. To obtain a minimal GD, a second step of vertex insertion is needed (crosses). The GD consisting of representatives marked in blue has minimal length and fulfills the local minimality criterion.

Lemma 4.3. *After two steps of inserting new vertices, all relevant cluster representatives start and end at vertices of the augmented data.*

Proof. Consider a relevant cluster representative τ starting and ending at arbitrary points along a trajectory and consider what makes τ relevant:

- **Case 1:** When extending τ , the distance to one of the cluster curves exceeds d or the distance to another trajectory, which is not in the cluster yet, falls below d .
- **Case 2:** When extending τ , the distance of one of the cluster curves σ to another trajectory, which not in the cluster, falls below d .

The event described in case 1 implies that a vertex v is inserted to τ in the first propagation step. The event in case 2 implies the insertion of a vertex w to σ and, since the distance of σ and τ is smaller than d , a vertex v is inserted to τ in the second propagation step. Thus, we can restrict τ to start at v .

Lemmas 4.2 and 4.3 imply:

Theorem 4.4. *After two steps of inserting new vertices, there always exists a minimal GD solution wherein edges correspond to restricted relevant cluster representatives when minimizing size. When minimizing length, this solution adds at most an additive error of $2de$.*

4.2. Construction of the subsets

For the SET-COVER construction, we use a sweep algorithm to determine relevant cluster representatives for each of the given trajectories. The idea is to extend each representative until it is relevant, report the corresponding cluster, and proceed with the endpoint of this representative as the starting point of the next representative.

With this approach, we ensure that we only obtain representatives of which a minimal selection covering the whole input data also fulfills the additional criteria stated in the definition. As we cut the representative every time when the part G^* as described in the problem definition changes, we obtain both: on the one hand, by giving each representative (corresponding subset) a unit weight, an optimal solution for the whole data can only be achieved when the solution for each part G^* is minimal. On the other hand, if we limit the selection to relevant representatives, we avoid artificial splits. Note that during the sweep algorithm, we only need to report clusters where all cluster curves (not only the representative) are restricted, i.e. they start and end at a vertex. Details on this remark can be found in (Buchin et al. 2018).

The algorithm described above is summarized in the following pseudocode. The correctness of Algorithm 1 follows from Lemma 4.2. For a given trajectory T of complexity N , $T_{a,b}$ for $a, b \in \{1, 2, \dots, N\}$, $a < b$, denotes the subtrajectory starting at vertex number a and ending at vertex number b .

Algorithm 1: Compute subsets for SET-COVER instance

Input k trajectories $\{T_1, T_2, \dots, T_k\}$ of complexity $N_j, j = 1, \dots, k$
 boolean variable *minimizingLength* # TRUE for minimal total edge length,
 FALSE for minimal number of edges
 empty list S % all computed subsets are added to S
 empty list R % all representatives (subtrajectories) of the clusters (subsets) are added to R
Output list R , list S

```

for  $j = 1, 2, \dots, k$  do
   $T \leftarrow T_j$ 
   $a \leftarrow 0$ 
   $b \leftarrow 1$ 
  repeat
    do
      compute  $c(T_{a,b})$ 
       $is\_irrelevant \leftarrow checkRelevance(T_{a,b}, T_{a,b+1}, c(T_{a,b}))$ 
       $b \leftarrow b + 1$ 
    while  $is\_irrelevant$  and  $b < = N_j$ 
    add all segments of curves in  $c(T_{a,b})$  as subset to  $S$ .
    Set the weight according to the minimality criteria:
    if minimizingLength then
       $w(c(T_{a,b})) \leftarrow length(T_{a,b})$ 
    else
       $w(c(T_{a,b})) \leftarrow 1$ 
    end
    Add  $T_{a,b}$  to  $R$  with the same index as the corresponding subset in  $S$ .
     $a \leftarrow b - 1$ 
  until  $b > N_j$ ;
end

```

Algorithm 2: checkRelevance

Input $T_{a,b}, T_{a,b+1}, c(T_{a,b})$

```

if all cluster curves start and end at vertex of the augmented input then
  compute  $c(T_{a,b+1})$ 
  if  $L_{T_{a,b}}$  equal to  $L_{T_{a,b+1}}$  then
    if  $G_{T_{a,b}}^-$  smaller than  $G_{T_{b,b+1}}^+$  then
      return FALSE
    else
      return TRUE ;
  else
    return FALSE
  end
else
  return TRUE ;
end

```

When solving the constructed SET-COVER instance, we only need to keep track of the indices of the chosen sets in S to select the corresponding representatives from R to build the GD. For each trajectory, we maintain a list L_τ of all trajectories which maintain subtrajectories within distance d to the current representative τ . Furthermore, we store the part G^* based on the last segment of $T_{a,b}$. If all cluster curves start and end at a vertex and if $T_{a,b}$ is a relevant cluster representative, we report $T_{a,b}$ and the cluster $c(T_{a,b})$ as one subset in S of the SET-COVER instance. Recall that is the case, if $|L|$ increases or decreases when we extend $T_{a,b}$ by one segment or if new cluster curves are added to G^* when extending $T_{a,b}$.

Computational Complexity For reporting the clusters, we use the same approach as in (Buchin et al. 2011). First, we compute the free-space diagram (FS-Diagram) for the input trajectories which can be done in $\mathcal{O}(k^2n^2)$ time. Note that the time complexity for computing the FS-Diagram only depends on the complexity of the input trajectories, which is at most n for each of the k trajectories.

Let $N = \max_{j=1,\dots,k} N_j$. For each trajectory, we have to perform at least N queries in case that only the whole trajectory is a relevant cluster representative and at most $2N$ queries as we have at most $2N$ different positions of a and b . Each query takes $\mathcal{O}(kNl)$ time (Buchin et al. 2011), where l is the number of segments of the cluster representative. The average of l per query is maximal, if the whole trajectory is a relevant cluster representative. In this case, we have: $l = \frac{1}{N} \sum_{j=1}^N j = \frac{N+1}{2} \in \mathcal{O}(N)$. Thus, the total query time sums up to $\mathcal{O}(kN^3)$. Now we determine the complexity of the additional steps of the algorithm. To obtain G^* , we have to compute the cluster represented by the last segment of the current representative, $\tau_{\text{last_segment}}$. In total, we have to perform $\mathcal{O}(kN)$ queries of length 1. Thus, the complexity is $\mathcal{O}(k^2N^2)$. Each cluster $c(T_{b-1,b})$ contains at most k cluster curves of complexity at most N . Thus, the complexity to compute $G^* = c(c(T_{b-1,b}))$ is $\mathcal{O}(kN^2)$ and for kN segments we have $\mathcal{O}(k^2N^3)$ in total. As the lists described in the algorithm have size at most k , we can compare them in $\mathcal{O}(k)$ time. Having $\mathcal{O}(kN)$ comparisons, the time complexity is $\mathcal{O}(k^2N)$ for these computations.

Theorem 4.5. *Given a GD instance, we can compute in $\mathcal{O}(k^2N^3)$ time a SET-COVER instance of size $|\mathcal{U}| = \mathcal{O}(kN)$ and $|\mathcal{S}| = \mathcal{O}(kN)$, the solution of which solves the GD instance.*

The theorem follows directly from the analysis above. A discussion about the complexity of a trajectory after two steps of vertex insertion and the computation time of the insertion can be found in (Buchin et al. 2018). Note that the time for computing the subsets described in the algorithms above dominates the time complexity of the preprocessing.

5. Group diagrams based on the equal- and similar-time distance

Next, we want to compute a GD based on equal-time distance as similarity measure. A path P within a group diagram is similar to an input trajectory T , if for any t in the domain of T the Euclidean distance $\text{dist}(P(t), T(t))$ is at most d .

The following observation is a direct consequence of the linear interpolation between two vertices of a trajectory: Given two piecewise-linear trajectories T_1, T_2 , with vertices at the locations corresponding to the timestamps $t_0, \dots, t_m$, then, if $\text{dist}(T_1(t_i), T_2(t_i)) \leq d$ and $\text{dist}(T_1(t_{i+1}), T_2(t_{i+1})) \leq d$, we have $\text{dist}(T_1(t), T_2(t)) \leq d$ for all $t \in (t_i, t_{i+1})$.

5.1. Segmentation

Exploiting this observation, we insert a sufficient number of timestamps and corresponding vertices additional to the input vertices to ensure that in between consecutive timestamps the pairwise equal-time distance of the trajectories does not change with respect to the threshold d . To this end, we insert a vertex to each trajectory using linear interpolation for each of the at most kn different timestamps given by the input. Subsequently, we consider only the common time intervals of all trajectories. Then, we compare all segments between two consecutive timestamps in a second step. Obviously, for computing a minimal GD, it is sufficient to use restricted representatives with vertices in V , where V is the set of vertices after the preprocessing.

Let $\overline{AB}$ and $\overline{CD}$ be two segments of different trajectories between two consecutive timestamps i and $i+1$. If $\text{dist}(\overline{AB}_t, \overline{CD}_t) \leq d$ holds for $t = t_i$ and $t = t_{i+1}$, the segments are at equal-time distance at most d for all $t \in (t_i, t_{i+1})$ and we do not need to insert any new vertices. If $\text{dist}(\overline{AB}_t, \overline{CD}_t) \leq d$ holds for t_i but not for t_{i+1} , the equation $\text{dist}(\overline{AB}_t, \overline{CD}_t) = d$ has exactly one solution t_s in (t_i, t_{i+1}) , and we insert a new vertex to all trajectories (if possible) at the corresponding locations at t_s (*split event*). Analogously, we calculate t_s and insert new vertices, if the inequality holds for t_{i+1} but not for t_i (*merge event*). Lastly, if the inequality does not hold for t_i nor for t_{i+1} , the equation $\text{dist}(\overline{AB}_t, \overline{CD}_t) = d$ has either no solution or exactly two solutions $t_{\min}$ and $t_{\max}$ in I . In the first case, we can conclude that the segments do not share a part where the equal-time distance is less than or equal to d . In the latter case, we obtain one merge and one split event between t_i and t_{i+1} . Again, we insert vertices to every trajectory at time $t_{\min}$ and $t_{\max}$.

Figure 9 shows the trajectories T_1 and T_2 of two entities moving along a similar route, but with different speed. The sampled positions at t_0 and t_1 for T_1 and at t_0 and t_2 for T_2 are depicted as discs. A vertex at t_1 is added to T_2 to simulate equal-time sampling and vertices at the timestamp where the distance of the entities equals d (at t_{new}) is added to both trajectories. The part where the two entities were traveling within equal-time distance is highlighted with thick, red edges.

Lemma 5.1. *The segmentation takes $\mathcal{O}(k^4 n \log n)$ time. After this process each of the k trajectories has $\mathcal{O}(k^3 n)$ vertices.*

Proof. Each comparison takes constant time. The stated complexity results from the number of comparisons which is $\mathcal{O}(k^3 n)$ and the time needed for one vertex propagation which is $\mathcal{O}(k \log n)$. $\square$

Figure 9. Equal-time segmentation of two trajectories.

5.2. Computing the GD

For computing the GD, we proceed in the following way: Between each two consecutive timestamps in V , we compute one subset for each segment which contains the indices of all other segments within equal-time distance at most d . The distance between the two segments is the maximum of the Euclidean distance between the two starting points of the segments and their two ending points.

Then, we solve the SET-COVER instance and report the segments that correspond to the selected subsets. When minimizing the total edge number of the GD, we have to ensure that the representation does not change when not necessary in terms of minimality. Otherwise, the GD consists of edges that could be concatenated. This can happen because the solution of the SET-COVER, in general, is not unique. To maintain one representation as long as possible, we check if the representation R_{old} between the previous two timestamps still represents all segments between the current two timestamps and if the size of R_{old} equals the size of the current solution. In this case, we maintain R_{old} and proceed with the next timestamp. This additional step is not necessary when minimizing the total edge length. Here, the sum of the length of a minimal length representation between a series of consecutive timestamps within a time frame from start t_s to end time t_e is at most the minimal length of a representation looking at the whole interval $[t_s, t_e]$ at once.

Lemma 5.2. *For each timestamp, we can compute the k sets of the SET-COVER instance in $\mathcal{O}(k^2)$ time.*

Lemma 5.1 and Lemma 5.2 imply the following theorem:

Theorem 5.3. *Given a GD instance using equal-time distance, we can compute in $\mathcal{O}((k^5 + k^4 \log n)n)$ time $\mathcal{O}(k^3n)$ SET-COVER instances each of size $|\mathcal{U}| = k$ and $|\mathcal{S}| = k$ the solution of which solves the GD instance.*

5.3. Similar-time distance

Equal-time distance may be too restrictive for some applications, for example, if there is a leader and followers within a stock, such that the followers move along the same route as the leader, but reach each position with a small delay. We introduce the similar-time distance as an additional similarity measure for such settings.

Definition 5.4. Given two trajectories T_1 and T_2 , where T_1 has size m and T_2 has size n . We say that T_1 has α -similar-time distance at most d to T_2 , if for all timestamps $t_i, i = 0, 1, \dots, n-1$ there exists $\hat{t}_i \in [t_i - \alpha, t_i + \alpha]$, $\hat{t}_{i+1} \in [t_{i+1} - \alpha, t_{i+1} + \alpha]$, $\hat{t}_i \leq \hat{t}_{i+1}$, with $\text{dist}(q_i, \hat{p}_i) \leq d$, $\text{dist}(q_{i+1}, \hat{p}_{i+1}) \leq d$, where $\hat{p}_i$ is the location of T_1 at time $\hat{t}_i$ and q_i is the location of T_2 at t_i , and for all $t \in [\hat{t}_i, \hat{t}_{i+1}]$ T_1 stays within a d -tube around $q_i q_{i+1}$.

Figure 10 shows the same two trajectories as Figure 9. The trajectory T_1 has α -similar-time distance to T_2 for any $\alpha \geq 20$ minutes. Between 00:00 and 00:50, the entities are moving within similar-time distance.

Figure 10. Illustration of α – similar-time distance.

For computing a GD with similar-time distance as similarity criterion, we use the same approach as for equal-time distance and only adapt the construction of the sets for the SET-COVER instance for each timestamp by comparing each segment with the segments of the other trajectory in the interval given by the allowed time shift.

In addition to the conditions of Definition 5.4, we can make the restriction that the length of the time interval between $\hat{t}_i$ and $\hat{t}_{i+1}$ does not fall below a given minimal value, for instance, a fraction of the length of the interval $[t_i, t_{i+1}]$. Thus, we still allow a time shift when two entities travel along a similar route, but additionally, demand that the duration of traveling a similar route does not differ too much. For instance, we exclude two trajectories to be considered similar, if one entity stops moving for a larger amount of time (e.g. a stopover of a goose) where the other entity continued moving.

Definition 5.5. Given two trajectories T_1 and T_2 , where T_1 has size m and T_2 has size n . We say that T_1 has α - β -similar-time distance at most d to T_2 , if for all timestamps $t_i, i = 0, 1, \dots, n-1$ there exists $\hat{t}_i \in [t_i - \alpha, t_i + \alpha]$, $\hat{t}_{i+1} \in [t_{i+1} - \alpha, t_{i+1} + \alpha]$, $\hat{t}_i \leq \hat{t}_{i+1}$, with $\text{dist}(q_i, \hat{p}_i) \leq d$, $\text{dist}(q_{i+1}, \hat{p}_{i+1}) \leq d$, where $\hat{p}_i$ is the location of T_1 at time $\hat{t}_i$ and q_i is the location of T_2 at t_i , and for all $t \in [\hat{t}_i, \hat{t}_{i+1}]$ T_1 stays within a d -tube around $q_i q_{i+1}$ and $\hat{t}_{i+1} - \hat{t}_i \geq \beta(t_{i+1} - t_i)$.

6. Experiments

In this section, we return to the motivating problem of finding a concise representation for one or more groups of migrating greater white-fronted geese and present experimental results by applying the algorithms on GPS tracking data of a family and of independent individuals of geese. For the migration of goose families, we use equal- and similar-time distance as similarity criteria and for the migration of independent individuals, we use the Fréchet distance as similarity criterion. We are interested in a concise (visual) representation and obtain such a representation by computing minimal GD for the input trajectories and by plotting the edges of the GD on a map. This representation highlights typical migration routes for the input set of the entities, in the sense that the movement of each entity is similar to at least one of these routes. Furthermore, we use the solution of the constructed SET-COVER instance to summarize the migration of a goose family answering different questions. For instance, in relation to the whole migration period, for how long was the family flying close together, with respect to a varying distance threshold? Beyond representation, the obtained migration characterizations can help to form novel research hypotheses. Additionally, we investigate the runtime and scalability by introducing additional artificial group members.

We used the Python programming language and the library *simplekml* to create GD which can be displayed on *google maps* or *OpenStreetMaps*. To solve the (weighted) SET-

COVER instance, we used the library *SetCoverPy*. All computations were performed on an Intel Pentium i5 2.4 GHz processor and 12 GB RAM.

6.1. Equal-time and similar-time distance

In this section, we consider one-tracked goose family which consists of both parents and two juveniles. The tracks were collected between March and June 2017 on their spring migration route from North Western Europe to the Russian Arctic. Positions were collected in half-hourly bursts of 10 GPS positions (one position every 2 s). Each position provides a horizontal error value, so we preprocessed the data by deleting all points with high error, i.e. > 15 m. Because the goose family splits at the second part of its travel, we investigated the tracks between Denmark and Latvia here which contain around 2000 GPS positions for each animal, see Figure 11.

The distance between two entities is computed based on their positions on the earth's surface, i.e. the great circle distance. A GD shows when a subgroup (or one single entity) separates from the rest of the group (or from a subgroup) and when a subgroup joins another subgroup. Looking at the data of the migrating goose family, we observe that these split and merge events only happened on a relatively small scale compared to the total geographic extent of the data. This observation confirms the hypothesis that a family of greater white-fronted geese possibly stays together during the whole migration period – which has not been experimentally proven up to now due to the lack of available data and analyzing tools. Applying the GD technique to the data of the goose family showed that the family stayed together with a varying distance between the family members which is mostly below 100 m. See Figure 12 for an illustration. Generally, if a family member completely splits from the rest of the family on a larger scale, it is unlikely that a reunion of this member with the rest of the family or parts of the family occurs. However, detecting and visualizing split and merge events on a small scale is an interesting application of the GD as it reveals how the structure of the family (the distance between the members), changes during the flight. We use the GD to investigate when and where the family was flying close together so that it can be represented by only one member and for which periods of time more than one representative is needed.

We computed GD based on equal-time and α -similar-time distance as similarity measure. For the goose family tracks, we determined the GD for distances $d =$

Figure 11. The trajectory of one goose family member between Denmark and Latvia. The focus shows one burst of sampling.

Figure 12. Representativity of the goose family for different distance thresholds and equal- and similar-time distance. The average number of representatives needed is the average size of the SET-COVER solution between two timestamps over the whole observation period. A value of 4 indicates that during the whole migration period the pairwise distance between the family members exceeds the distance threshold, whereas a value of 1 implies that in between each two consecutive timestamps there exists one entity e , such that the distance to e is lower than the given threshold for each family member.

3, 5, 10, 20, 40, 80, 160, 320, 640, 1280 meters. Due to the wingspread of the geese, a distance lower than 3 m is unlikely to be observed. Doubling the distance in each step is motivated by the observation that the GD changes the most for small distances because for most of the time the maximum distance between two entities is small, mostly less than 40 m. However, this is a data set specific insight. For each distance, we set the allowed time shift to $\alpha = 10$ seconds when computing the GD with α -similar-time distance as a similarity measure. This parameter was experimentally chosen. For larger time shifts, only the computation time increases without a significant difference of the GD compared to the 10-s time shift.

In Figure 12, we compare the decrease of the average number of representatives needed for increasing distance thresholds to the increase of time when one representative is sufficient to represent the whole family for equal ($\alpha = 0s$) and similar-time ($\alpha = 10s$) distance. We observe that for small distances, allowing a time shift of 10 s has a greater impact on the average number of representatives needed than doubling the distance. For instance, we look at the distance threshold values of 10 and 20 m. For equal-time distance the percentage of time during which the whole family is represented by one representative increases from 38% to 44% when doubling the distance threshold from 10 to 20 m. When allowing a time shift of 10 s for a distance threshold of 10 m, one representative is sufficient for 51% of the timestamps. Analogous observations can be made for the average number of representatives and for all other distance thresholds below 40 m.

Figure 13 illustrates how allowing a time shift while maintaining the distance threshold leads to a smaller diagram in terms of the number of representatives needed. In this part of the GD, two representatives suffice when a time shift is allowed whereas for equal-time distance four representatives are needed.

As distance increases, it becomes the dominating parameter for the size of the GD. For large distance thresholds, the impact of an allowed time shift is much smaller than for

Figure 13. No time shift allowed (left) and a time shift of 10 s (right) for a distance threshold $d=10\text{m}$. The trajectory of each family member is shown in a different color. That is, in the left figure, each entity needs its own trajectory as a representative, whereas in the right figure, the blue and the green trajectories represent the movement of all four family members. The thickness of the subtrajectory τ of an entity e depends on the number of entities that are within a distance of 10 m to e during the lifespan of τ .

small distance thresholds. This can also be seen in Figure 13 as the difference of the values of the red and blue curves decrease with an increasing distance threshold.

Most likely, the reason for these observations is the formation of the flock while flying. If the entities of the flock are flying in a V-formation or in a single file, two entities are represented by one representative even if their distance is larger than the given threshold when allowing a small time shift. The impact of a time shift on the size of the GD would be less, if the birds were flying next to each other rather than behind each other (in a single file or a V-formation).

6.1.1. Migration over water and solid ground

During the migration period, one can observe that when the family was flying over surfaces of water, they tended to fly with greater pairwise distance from each other than while flying over solid ground. One example of this phenomenon is shown in Figure 14 for an equal-time distance of 160 m.

Figure 15 shows the difference in the number of representatives needed for flying over solid ground and for flying over water. One interesting observation is that the values differ most between 10 and 100 m.

6.1.2. Runtime and scalability

The equal-time algorithm consists of three steps. The computation of the event times and the vertex insertion, the construction of the SET-COVER instances, and the process of solving

Figure 14. Family members split when flying over a lake and merge again after passing the lake.

Figure 15. Average number of representatives needed to represent the whole family for different distance thresholds. This number indicates how much the family members are separated. The figure shows the difference in this value while flying over land and over water.

the SET-COVER instances. The runtime of all three steps highly depends on the number of split- and merge events in the given data.

To evaluate the sensitivity of our algorithms for a growing group size, we generate artificial trajectories based on the given trajectories of the family members. We use two different approaches here. First, we generate two random numbers r_1 and r_2 for each trajectory τ and add r_1 and r_2 to the coordinates of each vertex. Second, we generate two random numbers at each vertex separately and add the numbers to the corresponding coordinates. The range of r_1 and r_2 is between -0.0001° and $+0.0001^\circ$ which results in the shift of each vertex within a circle of approximately 10-m radius. Note that using the first approach, the whole trajectory is shifted in parallel to the original whereas the second approach produces zig-zag trajectories around the original trajectory which leads to a higher number of events in comparison to the first approach, see Figure 16 for an example.

Using these two ways of generating data, we want to emphasize that the scalability of the algorithm highly depends on the given data, see Figure 17. To take randomness of the generated data into account, we run each computation twice and take the average value of the measured runtime and the number of events. We thereby note that these values barely differ between two runs of the algorithm for the same input values. The correlation

Figure 16. Generating a new trajectory based on a given trajectory (thick line) by shifting each vertex equally in north-south and in the east-west direction (a) or by shifting each vertex independently (b).

Figure 17. The runtime of the algorithm using the first data generation type (left) (shifting each vertex independently) and the second type (right) (shifting the whole trajectory in north-south and east-west direction). Both data were generated based on the identical input data set.

Figure 18. The number of events using the first data generation type (left) and the second type (right).

of the type of data generation, the number of events and the runtime of the algorithms is illustrated in Figures 17 and 18.

For a comparison of the runtime for computing the events and the construction and solution of the SET-COVER instances, see Figure 19 (the first type of data generation was used here). Note that the construction of the SET-COVER instances dominates the runtime in the lower plot since in this step, we need to compare all pairs of entities. However, the SET-COVER instance computed between each two consecutive timestamps is relatively small; it consists of exactly k subsets. Therefore, the runtime of the greedy algorithm which computes an approximate solution of the SET-COVER instance is significantly smaller than the construction time. For a quantitative evaluation, see Table 1. The table shows the runtimes for a data set of eight entities obtained by the first type of data generation from the input data.

Figure 19. Runtime for computing the events including vertex propagation (upper plot) and for constructing and solving the SET-COVER instances (lower plot).

Table 1. Runtimes of the two steps of the algorithm.

Distance (m)	Construction time (s)	Solving time (s)	Total runtime (s)
3	35.9	0.9	36.8
5	52.0	1.0	53.0
40	4.9	0.1	5.0
80	3.1	0.0	3.1
460	0.5	0.0	0.5

Another parameter affecting the runtime is the allowed time shift when using similar-time as a similarity measure. Let $\tau(i, j)$ be the current cluster representative starting point at time $t = i$ and endpoint at time $t = j$. Let a be the allowed time shift and $\tau(i, j)$ the trajectory of entity e_1 between the timestamps i and j and let σ be the trajectory of entity e_2 . Clearly, the computation time increases if a increases, because the algorithm needs to traverse a longer subtrajectory, namely $\sigma(i - a, j + a)$, to examine whether e_1 and e_2 are within similar-time distance at most d between the timestamps i and j . See Figure 20 for an illustration.

6.2. Fréchet distance

In this section, we show that GDs can be used to extract representatives, i.e. typical migration routes from a data set of migration trajectories. To this end, we computed GDs with the Fréchet distance as a similarity criterion. Recall, that for the Fréchet distance only the order of the time-stamped locations is taken into account and not the specific time-stamp. Therefore, we can compare entities migrating during different periods of time. We consider two different settings. First, we compute GDs for one individual and its migration trajectories during several migration periods. Second, we consider several independent individuals for one specific migration period. For the goose family in the previous section, we computed GDs for small distance thresholds up to several hundred meters, as the family was migrating together and we were only interested in the varying separation of the

Figure 20. Measured runtime for computing a GD for different distance thresholds and values of allowed time shift.

family members. In this section, we want to obtain a concise representation of different migration routes. To this end, we use distance thresholds between 50 and 300 km.

6.2.1. Implementation and data description

We use the Python libraries *Shapely* and *LatLon* to compute the segmentation of the input data as described in Section 4.1. *Shapely* supports geometric operations such as projections of points onto line segments and the computation of the intersections of polygonal lines. With *LatLon*, we are able to perform parallel shifts of a line segment on the earth surface for a given distance d . In our implementation of the Fréchet distance, we use the Vincenty's formula provided by the library *geopy* for all point-to-point distance calculations.

We want to compare and to summarize the migration routes on a large scale, that is with a distance threshold between 50 and 300 km. Therefore, we are not interested in the movement during stopovers or during a burst of GPS positions. Moreover, to satisfy the memory constraints of the implementation (see the runtime analysis in Section 6.2.3), it is necessary to upper bound the resolution of the migration routes, i.e. the complexity of the trajectories. We have manually chosen a minimum distance of 90 km between two consecutive stored locations. By doing so, we exclude the stopovers and upper bound the positions for each burst but maintain a sufficiently fine-grained resolution of the migration routes for the distance thresholds we consider in the evaluations below.

Each relevant cluster (subset of the SET-COVER instance) is weighted by the relative length of its representative to the total length of all migration routes of the input and we solve the resulting $\text{WEIGHTED-SET-COVER}$ instance to obtain the GD with minimal total edge length. The length of each representative is approximately computed by projecting the longitudes and latitudes to the Mercator web with the library *PyProj* and by using the built-in length function of the resulting *Shapely-LineString* class instance of the projected subtrajectory.

In Figures 21-23, the GPS-tracks of the geese are plotted in black whereas the edges of the computed GDs are shown in red.

Figure 21. GDs (red) for spring migration between 2014 and 2016 (black) for a distance threshold of 50 km (upper left), 100 km (upper right) and 250 km (lower figure).

Figure 22. GDs (red) for spring migration in 2014 of three individuals (black) for a distance threshold of 100 km (left figure) and 300 km (right figure).

6.2.2. Group diagrams

One Individual – Multiple Years Three GDs representing the spring migration between 2014 and 2016 of one individual (id:711) of the study *LifeTrack Geese IG-RAS MPIO ICARUS* (data set maintained by movebank.org) are shown in Figure 21. The GD provides compact

Figure 23. GD (red) for spring migration in 2014 of eight individuals (black) for a distance threshold of 300 km. The thickness of the GD edges reflects the number of entities represented.

representations of the three migration routes and outlines similarities and differences in the migration periods for different distance thresholds. Note that to the best of our knowledge there are no data sets available for migrating geese with an observation period of more than 3 years.

In order to make the plots more readable and understandable, we connected the computed edges of the GD with additional straight-line, dark red-colored edges, if the endpoints of two edges are within the given distance threshold.

The distance thresholds have been chosen experimentally such that the three GDs significantly differ from each other. For a distance threshold of 50 km, two representatives are needed to capture all data, except for the part in northern Germany, the Finish Bay, and the Barents Sea where one representative is sufficient to represent the migration routes for all 3 years. As one can expect, the routes are similar at the beginning of the migration (Northern Germany) and close to the breeding grounds (Barents Sea). The fact that during all 3 years, the goose was migrating along a similar route along the Finish Bay might implicate that alternative routes with the same east-west expansions are less advantageous in terms of food supply or hunting. For a distance threshold of 100 km, one representative is sufficient except for the migration over Lithuania and Latvia. Finally, for a distance threshold of 250 km, the GD consists of only one representative, that is all the spring migration routes between 2014 and 2016 are inside a 250 km-tube around this representative. This example shows how a concise visual representation depending on the specific question can be obtained. If one is interested in extracting (and plotting)

representative migration routes which show spatial differences between the years on a large scale, one can choose a large distance threshold. With a decreasing distance threshold, the size of the GD increases, revealing spatial differences between the migration years on a smaller scale.

Multiple Individuals – One Year We can also use the GD framework to summarize the different migration routes for multiple individuals during one migration period. In [Figure 22](#), we present two GDs of three individuals from the study *LifeTrack Geese IG-RAS MPIO ICARUS* (data set maintained by [movebank.org](#)) for the observation period from February to June 2014. Again, the size of the GDs and the placement of the edges of the GDs reflect the similarity of the migration routes of the individuals for different distance thresholds. For the lower distance threshold (100 km), the migration routes can be summarized as follows: In the first part, all individuals are migrating inside a 100-km-tube of the migration route of one representative individual. Then, the routes split, once in eastern Poland and once in Lithuania. Finally, two merges occur at the coastline of the White Sea and at the coastline of the Barents Sea. A distance threshold of 300 km results in a GD which consists of only two different representative migration routes, one northern route along the coastline of the Baltic Sea and the White Sea and one southern route passing Ukraine and the midland of western Russia. Again, these distance thresholds have been chosen experimentally to show two significantly different GDs.

The given examples illustrate the idea of finding a concise visual representation for multiple tracks of moving objects, i.e. to extract typical routes for different distance thresholds and to avoid overplotting, especially for a small Fréchet distance between the input trajectories. However, computing such a representation is more interesting for groups of larger size. [Figure 23](#) shows a GD of the spring migration in 2014 for a group which consists of eight individuals from the study *LifeTrack Geese IG-RAS MPIO ICARUS* for a distance threshold of 300 km. This example also illustrates how the GD can be used to identify outliers, as the representative along the coastline captures the vast majority (seven out of eight) individuals whereas the representative along the midland of Europe captures the migration of only one individual. Due to memory constraints, we skipped the segmentation steps here and performed the clustering and the computation of the relevant clusters based on the input vertices solely. The possible errors of this approach, which are discussed in [Section 4.1](#), do not seem to play an important role here because the distance threshold is around three times higher than the resolution of the tracks. Hence, for the splits and merges on this large scale, the input data is already sufficiently fine segmented.

Note that the algorithm additionally provides the *representation strength* of each edge of the GD (see remarks below Definition 2.1) at no extra cost. Thus, when adjusting the thickness of each edge accordingly, the GD reflects that most individuals were migrating along the coastline and only one was following a route more towards the midland of Eastern Europe and Russia.

6.2.3. Runtime

We briefly summarize the runtime of the computations in this section. All experiments were performed twice and we show the average runtimes in the tables.

[Table 2](#) shows the number of vertices (Size) after two segmentation steps and the runtime of the different steps of the algorithm for all three considered distance thresholds. The runtime is broken down into the time needed to run the segmentation (Segm.),

Table 2. Runtime to compute the GD: one individual – multiple years.

Data		Computation time (s)				
Dist(km)	Size	Segmentation	Setup	Construction	Solving	Total
50	591	8.9	69.0	2.6	2.2	82.7
100	766	9.9	122	5.9	1.0	138.8
250	859	11.3	243.9	14.3	1.1	270.6

the time to compute all the data structures needed for the clustering (Setup), the computation of all relevant clusters and cluster representatives which form the SET-COVER instance (Constr.), and the runtime of solving the instance (Solve). The setup consists of the computation of the labeled augmented *Free-Space-Diagram*. This data structure allows to search for cluster curves with Fréchet distance at most d to a given representative. For more details see clustering.

Table 3 summarizes the runtimes to compute the GDs of three individuals for one migration period. The first two columns show the number of the vertices before (Input) and after the segmentation steps (Augmen.) for three different resolutions (Res.), which is the minimal distance between two input points.

Further experiments showed that the limit of the implementation in terms of the data size is roughly 1200 vertices after the segmentation step. Therefore, the computation of the GD fails for the distance threshold of 100 km and a resolution of 30 km, see Table 3. For a data size of 1000 vertices, we need to store up to one million float numbers in the augmented *Free-Space-Diagram* which requires 12 GB storage on our system. In order to perform experiments with a larger input size, the implementation would need to be more memory efficient as described in clustering. The runtime is dominated by the computation of the data structures needed for the clustering as well as by the computation of the relevant clusters.

6.3. Discussion of possible errors

GPS accuracy is crucial for the computation of GDs. In particular, relative GPS accuracy is crucial for small distance thresholds. We use the absolute accuracy which is provided as a real number within the data set as a criterion for selecting the locations for computing

Table 3. Runtime to compute the GDs: 1 year – multiple individuals for a distance threshold of 100 km (upper table) and 300 km (lower table).

Data			Comp. Time (s)				
Res. (km)	Input	Augmen.	Segm.	Setup	Constr.	Solving	Total
100	115	356	5.4	23.5	0.9	0.7	30.5
60	179	512	10.3	51.4	1.7	1.1	64.5
40	252	798	20.8	139.0	4.8	1.7	166.3
30	350	1589	62.6	MemErr	–	–	–

Data			Comp. Time (s)				
Res. (km)	Input	Augmen.	Segm.	Setup	Constr.	Solving	Total
100	115	484	7.5	60.3	2.7	0.9	71.4
60	179	731	14.9	161.7	6.9	2.4	185.9
40	252	1122	32.6	466.1	15.0	5.8	519.5

a GD (high numbers refer to inaccurate values). It has often been stated that relative accuracy is lower than absolute accuracy, so we assume that our data are sufficiently accurate, if we do not consider the (small) number of inaccurate values. For the Fréchet distance experiments, GPS accuracy can be neglected for the large distance thresholds we considered. Another possible source of error that more likely affects our results is the necessary interpolation between two given timestamps which only gives a rough estimation of the position in between two timestamps. For the purpose of computing a GD, a more uniform sampling rather than dense bursts with large time gaps would lead to better results.

7. Conclusion

The GD framework presented in this paper provides a method for a concise representation of the data of moving entities forming one or several moving groups. In particular, a concise visual representation is obtained by plotting the GD representatives onto a map. This representation shows typical routes for the entities in the data set, that is each individual movement is represented by at least one route in the representation. Note that the plotted GD is a geometric graph. Thus, for instance, one can apply graph distance measures to two GDs obtained from different data sets to reveal similarities and differences in the movement patterns of different groups. The GD can be adapted to several different types of movement and research areas by specifying the similarity measure and the minimality criterion accordingly. We presented a generic algorithmic approach to compute a GD and provided a detailed theory on how to compute the GD for the equal-time distance, the similar-time distance, and the Fréchet distance as similarity measures. With these distances, one can compute GDs for groups where all individuals are moving simultaneously or at different periods of time.

We find that our experimental results on data of migrating geese give an accurate picture of the movement for different types of groups of geese (family, one individual for multiple periods, multiple independent individuals). On the one hand, it is sufficiently information preserving, that is each individual movement is represented by at least one route consisting of representatives computed by the GD algorithm. On the other hand, the number of or the total geographic length of the representatives is minimized. Hence, the input data are represented in a maximally concise way.

Applying the GD framework on migration data of a family of greater white-fronted geese, we could confirm the hypothesis that a family can stay together during the whole migration period. The example of different GD patterns between bursts of flight over land vs. water reveals the usefulness of the obtained representation of the data for further analysis and ecological research. The GD of the goose family revealed that the family was more separated when flying over water than when flying over land. An observation, which motivates further experiments to confirm and to explain this hypothesis.

In the future, it would be interesting to apply the GD framework to more data sets such as vehicle tracking or hiking data. Data sets of groups moving together where several individuals are tracked simultaneously or data sets of one individual traveling a similar route during multiple periods of time have been quite rare up to now, but are becoming more common. Hence, we hope to perform more experiments with data from different fields. As the framework allows to specify the distance measure, one could elaborate additional constraints (e.g. speed of the entities) by applying a distance measure that captures such constraints.

Acknowledgments

We are grateful to Gerard Müskens and the Dutch Association of Goose Catcher for help with catching and tagging goose families. AK acknowledges funding from the DLR (ICARUS directive).

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the Deutsche Forschungsgemeinschaft (DFG) [BU 2419/3-1].

Notes on contributors

Maike Buchin is a professor at the Ruhr University in Bochum, Germany. Her research interests are algorithms and computational complexity of geometric problems, in particular problems arising in movement analysis.

Bernhard Kilgus is a PhD student in the Department of Mathematics at the Ruhr University in Bochum, Germany. His research activity focuses on geometric algorithms, in particular algorithms for movement analysis.

Andrea Kölzsch is a Postdoctoral Researcher at the Max Planck Institute for Animal Behavior in Radolfzell, Germany. Her research focuses on the ecology and behaviour of migratory Arctic geese under global change, for which she uses high-resolution GPS tracking and remote sensing.

References

- Ahmed, M., et al., 2015. *Map construction algorithms*. Switzerland: Springer.
- Ahn, H., et al., 2016. A middle curve based on discrete fréchet distance. In: *LATIN 2016: Theoretical informatics - 12th Latin American symposium proceedings*, 11-15 April 2016 Ensenada, Mexico, 14–26.
- Alt, H. and Godau, M., 1995. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry and Applications*, 5 (1), 75–91. doi:10.1142/S0218195995000064.
- Baltzer, O., et al., 2008. Olap for trajectories. In: *Database and expert systems applications*. 340–347.
- Benkert, M., et al., 2008. Reporting flock patterns. *Computational Geometry*, 41 (3), 111–125. doi:10.1016/j.comgeo.2007.10.003.
- Buchin, K., et al., 2010. Median trajectories. In: *Proceedings of the 18th annual European conference on algorithms: Part I*, ESA'10, Liverpool, UK, 463–474.
- Buchin, K., et al., 2011. Detecting commuting patterns by clustering subtrajectories. *International Journal of Computational Geometry and Applications*, 21 (3), 253–282. doi:10.1142/S0218195911003652.
- Buchin, K., et al., 2015. Trajectory grouping structure. *Journal of Computational Geometry*, 6 (1), 75–98.
- Buchin, K., et al., 2016. Compact flow diagrams for state sequences. In: *Experimental algorithms*. St. Petersburg, Russia: Springer Int. Publishing, 89–104.

- Buchin, M., Kilgus, B., and Kölsch, A., 2018. Group diagrams for representing trajectories. In: *Proc. 11th ACM SIGSPATIAL Int. Workshop on Comput. Transportation Science*, Seattle, WA, 1–10.
- Clark, B.N., Colbourn, C., and Johnson, D., 1990. Unit disk graphs. *Discrete Mathematics*, 86 (1–3), 165–177. doi:10.1016/0012-365X(90)90358-O.
- Ester, M., et al., 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In: *Proceedings of the second international conference on knowledge discovery and data mining, KDD'96*, Portland, OR, 226–231.
- Gudmundsson, J., Laube, P., and Wolle, T., 2012. *Computational movement analysis: springer handbook of geographic information*. Berlin, Heidelberg: Springer.
- Gudmundsson, J. and van Kreveld, M., 2006. Computing longest duration flocks in trajectory data. In: *14th ACM SIGSPATIAL symp. on advances in geographic information systems*, Arlington, VA, 24–35. doi:10.1016/j.jhsb.2006.09.007
- Gudmundsson, J., van Kreveld, M.J., and Speckmann, B., 2007. Efficient detection of patterns in 2d trajectories of moving points. *GeoInformatica*, 11, 195–215. doi:10.1007/s10707-006-0002-z
- Huang, Y., Chen, C., and Dong, P., 2008. Modeling herds and their evolvments from trajectory data. In: *Geographic information science*. Berlin, Heidelberg: Springer, 90–105.
- Jeung, H., et al., 2008. Discovery of convoys in trajectory databases. In: *Proc. VLDB Endowment*. vol. 1, Auckland, New Zealand, 1068–1080.
- Kostitsyna, I., et al., 2015. Trajectory grouping structure under geodesic distance. In: *Proc. 31st annual symposium on comput. geometry*, Eindhoven, Netherlands, 674–688.
- Li, Z., et al., 2010. Swarm: mining relaxed temporal moving object clusters. *Proceedings of the VLDB Endowment*, 3 (1–2). doi:10.14778/1920841.1920934.
- van Goethem, A., et al., 2016. Grouping time-varying data for interactive exploration. In: *Proc. 32th annual symposium on computational geometry*. vol. 51, Boston, MA, 61:1–61: 16.
- van Goethem, A., et al., 2017. Multi-granular trend detection for time-series analysis. *IEEE Transactions on Visualization and Computer Graphics*, 23 (1), 661–670. doi:10.1109/TVCG.2016.2598619.
- van Kreveld, M., et al., 2016. A refined definition for groups of moving entities and its computation. In: *Proc. 27th Int. Symposium on Algorithms and Computation*, Sydney, Australia.
- van Kreveld, M.J., Löffler, M., and Staals, F., 2015. Central trajectories. *CoRR*, abs/1501.01822.
- Vieira, M.R., Bakalov, P., and Tsotras, V.J., 2009. On-line discovery of flock patterns in spatio- temporal data. In: *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems, GIS '09*, Seattle, WA, 286–295. doi:10.1016/j.antiviral.2009.09.016
- Wiratma, L., Löffler, M., and Staals, F., 2018. An experimental comparison of two definitions for groups of moving entities. In: *Proc. 10th Int. conference on geographic information science*, Melbourne, Australia, LNCS. Springer.