Group Diagrams for Representing Trajectories

Maike Buchin[1] **and Bernhard Kilgus**[2]

1 **Faculty of Computer Science, TU Dortmund,** `maike.buchin@tu-dortmund.de`
2 **Department of Mathematics, Ruhr University Bochum,** `bernhard.kilgus@rub.de`

Abstract

We propose the group diagram as a representation for multiple trajectories representing one or several moving groups. Given a distance threshold, a similarity measure and a minimality criterion a minimal group diagram is a minimal representation of the groups maintaining the spatio-temporal structure of the groups' movement. We give hardness results and approximation algorithms for computing several variants of the group diagram.

1 Introduction

A moving object, called *entity*, is described by its location at n time stamps and a linear interpolation inbetween each two consecutive time stamps. The corresponding trajectory therefore is a polygonal line. Given k trajectories, each of complexity n forming one or several (overlapping, i.e., splitting and merging) groups we introduce the *group diagram* as a means of compactly representing these groups. We propose the following general definition:

▶ **Definition 1.1.** A group diagram (GD) is a geometric graph with vertices augmented by a temporal component, that 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. We say a group diagram is minimal if it is minimal in size, either with respect to its number of edges or the total length of edges.

We consider GD which are built from the input trajectories, i.e., edges of the GD are represented by subtrajectories of the input and two edges share a vertex if the endpoints of the corresponding subtrajectories are within distance d from each other. Endpoints of edges with no d-distance neighbor have degree one. Vertices in the graph are hence embedded as the set of end points of incident edges. We will use such graphs in the following. Note that we could transform these into planar embedded graphs, for instance by choosing and connecting to the midpoint of the point set of a vertex.

As similarity measure we consider three popular measures on trajectories: the Fréchet distance, equal-, and similar-time distance. Figure 1 illustrates several trajectories. The subtrajectories forming a minimal GD for the given trajectories are highlighted in red.

Minimizing the number of edges or their total length seems intuitively reasonable. However both can lead to strange effects illustrated in Figure 2 which shows two simple examples (one or two trajectories of complexity 1). In the left figure the input consists of a single trajectory of length $6d$ and the GD with minimal length consists only of the red points, which is a bad representation of the movement. In the right picture, the GD minimizing the number of edges consists of the two input trajectories, although we would like the common movement between A and B to be represented by only one representative. To prevent these effects we make the following further requirements.

- When minimizing number of edges, we require that as much as possible is jointly represented. Given a subtrajectory τ of an edge of a GD $\mathcal{G}$, let $c(\tau)$ denote all subtrajectories

34th European Workshop on Computational Geometry, Berlin, Germany, March 21–23, 2018.

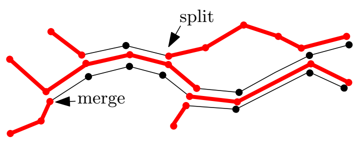
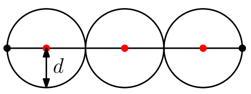


Figure 1 Illustration of GD.



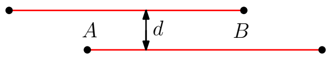


Figure 2 unintuitive GD with minimal edge length (left) and minimal number of edges (right).

of the input trajectories within distance at most d to τ and let $G^*_\tau := c(c(\tau))$ denote the union of all subtrajectories within distance at most d to a curve in $c(\tau)$. Furthermore, we define $A_\tau := \mathcal{G} \cap G^*_\tau$. We now demand that for each subtrajectory τ the resulting set A_τ is a minimal representation for $c(A_\tau)$.

- When minimizing the total 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 contracted, i.e., substituted by a subgraph of smaller size (but possibly larger length).

Related Work Two related notions to the GD are the grouping structure and flow diagrams. The grouping structure is the unique graph representing all density-connected groups traveling at equal-time [6]. A flow diagram is a minimal (in the number of vertices) diagram representing segmentations of all input trajectories. In a flow diagram nodes represent criteria and edges transitions between criteria [3]. The grouping structure is a specialization of the GD, which uses the equal-time distance, and density connectedness as inner group distance. The flow diagram can be seen as generalization of the GD (after switching between vertices and edges) where criteria are more general than small distance of the trajectories. Computing a GD with Fréchet distance as distance measure is also highly related to map construction algorithms, where the goal is to determine the underlying network of a set of trajectories [1]. Similar modeling choices (edges, similarity measure, and minimality condition) occur in the problem of finding a representative (median, middle, ...) trajectory of a set of similar trajectories.

Complexity Analysis By a reduction from the known NP-complete DOMINATING-SET problem for a grid graph [7] we can show that the decision problem for GD is NP-complete for all variants we consider.

▶ **Theorem 1.2.** *Given an integer l, deciding whether there exists a GD of size l is NP-complete 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.*

In the following two sections we give approximation algorithms and their experimental evaluation on a real data set. Due to space limitations many details, in particular proofs, are omitted in this extended abstract, and will be given in the full version.

2 Approximation Algorithms

Our approach is based on a natural formulation of the problem as a SET-COVER instance, which we first build and then solve approximately. To do so, we use the following concepts. A *cluster* is a set of trajectories called *cluster curves* that are all similar (under some similarity measure) to one *representative* of the cluster. Each edge in a GD can be identified with a representative of a cluster of subtrajectories from the input. We first detect all *relevant cluster representatives* and then we select a minimal set of these clusters where the union covers the complete input. Thus our 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 segment the trajectories such that we only need to consider subtrajectories starting and ending at vertices of the segmentation as cluster representatives.

As we construct and approximately solve a SET-COVER instance we obtain approximation algorithms in both cases, minimizing the total size of the GD and minimizing the length. When minimizing length for the Fréchet distance we additionally make a small additive error for each edge of the GD, see Lemma 2.2.

To take the two different minimality criteria into account we can formulate a weighted SET-COVER problem where we assign a weight to each representative (subset) depending on which minimality condition we choose: unit weight for number of edges and length of representative for edge length. Next we show how to implement this approach for the different distance measures.

2.1 Fréchet Distance

Recall that the Fréchet distance between two curves τ and σ is defined as the infimum over all reparameterizations α and β of $[0,1]$ of the maximum over all $t \in [0,1]$ of $\|\tau(\alpha(t)) - \sigma(\beta(t))\|$ [2]. To compute a minimal GD with Fréchet distance as similarity measure we use a sweep algorithm with two moving points a and b along every trajectory and report all *relevant* clusters represented by the subtrajectory between the current positions of a and b as described in [4].

▶ **Definition 2.1.** 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 no other trajectory not in the cluster enters a d-tube around one of the cluster curves. If a cluster representative cannot be extended in such a way the representative and the corresponding cluster are *relevant*.

▶ **Lemma 2.2.** *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.*

Note that this additive error is tight: Consider an input setting of trajectories of complexity one and length greater than $2d$, where always two trajectories are congruent and the pairs are within distance greater than d. Here, only the whole trajectories are relevant and therefore each representative (whole trajectory) is $2d$ longer than a minimal representative (middle part with distance d to endpoints).

When each segment has length greater than $4d$, which is the case in our experiments, we have a multiplicative error of at most 2 when using only relevant representatives.

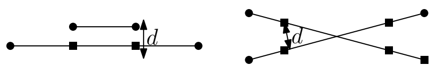


Figure 3 Inserting new vertices. The vertices from the input are shown as disks whereas the newly added ones are marked as squares.

Segmentation First we observe that using the segmentation given by the input vertices does not suffice for a minimal representation, e.g., in the case of parallel lines with different starting points. To obtain a sufficiently fine partition of the trajectories we consider two different triggers for inserting a new vertex. Firstly, for every vertex v of the input data we add a vertex to every segment which has distance to v less than or equal to d (see Figure 3a) at the point along the segment where the distance to v is minimal (type 1). Secondly, we add a new vertex if the distance between two segments is less than d for the first time and if the distance exceeds d again (type 2) (see Figure 3*b*).

▶ **Lemma 2.3.** *After two steps of inserting new vertices all relevant clusters start and end at vertices.*

From Lemma 2.3 it follows that we can use the vertices of the trajectories after two steps of vertex insertion as the event points of the sweep algorithm. For each trajectory τ we move b to the right until the representative is relevant and all cluster curves of the corresponding cluster start and end at vertices. Then we report this cluster as one subset of the Set-Cover instance, set a to the position of b and proceed like this until we reach the end of τ.

▶ **Theorem 2.4.** *Let N be the complexity of a trajectory after two steps of vertex insertion. Given a GD instance using Fréchet distance 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.*

▶ Remark. The value N is in $O(k^2C^2n)$, where C is a constant bounding the number of intersections of one segment with all segments of the input (see the full version for details). Note that this is linear in the dominating parameter n, since $k \ll n$.

2.2 Equal- and Similar-Time Distance

Next we want to compute a GD based on equal-time distance as similarity measure [5]. A path P within a group diagram is similar to an input trajectory τ if for any t in the domain of τ the Euclidean distance $dist(P(t), \tau(t))$ is at most d. The following observation follows directly from the linear interpolation between two vertices of a trajectory.

▶ Remark. Given two piecewise-linear trajectories τ_1, τ_2 with vertices at the locations corresponding to the time stamps $t_1, ..., t_m$. Then if $dist(\tau_1(t_i), \tau_2(t_i)) \leq d$ and $dist(\tau_1(t_{i+1}), \tau_2(t_{i+1})) \leq d$ we have $dist(\tau_1(t), \tau_2(t)) \leq d$ for all $t \in (t_i, t_{i+1})$.

Segmentation Using this observation we insert a sufficient number of time stamps and corresponding vertices additional to the input vertices to ensure that between consecutive time stamps the pairwise equal-time distance of the trajectories does not change with respect to threshold d. We do this by simulating equal-time distance first, i.e., inserting (by interpolation) a vertex to each trajectory for the at most kn different time stamps. Subsequently, we consider only the common time interval of all trajectories. Then we compare all segments between two consecutive time stamps in a second step.

Let $\overline{AB}$ and $\overline{CD}$ be two segments of different trajectories between two consecutive time stamps i and $i+1$. If $dist(\overline{AB}_t, \overline{CD}_t) \leq d$ 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 $dist(\overline{AB}_t, \overline{CD}_t) \leq d$ holds for i but not for $i+1$ the equation $dist(\overline{AB}_t, \overline{CD}_t) = d$ 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 $dist(\overline{AB}_t, \overline{CD}_t) = d$ 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 $t_{\max}$.

▶ Lemma 2.5. *The segmentation takes $\mathcal{O}(k^4 n \log n)$ time. After this process each of the k trajectories has at most k^3n vertices.*

Computing the GD For computing the GD we proceed in the following way. Between each two consecutive time stamps 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 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 which 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 time stamps 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 time stamp. This additional step is not necessary when minimizing the total edge length as the sum of the length of a minimal length representation between a series of consecutive time stamps 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 2.6. *For each time stamp we can compute the k sets of the* Set-Cover *instance in $\mathcal{O}(k^2)$ time.*

▶ Theorem 2.7. *Given a GD instance using equal time distance, we can compute in $\mathcal{O}((k^5 + k^4 \log n)n)$ time $\mathcal{O}(k^3 n)$* Set-Cover *instances each of size $|\mathcal{U}| = k$ and $|\mathcal{S}| = k$ the solution of which solves the GD instance.*

Similar-time Distance Equal-time distance may be too restrictive for some applications, for example for entities which travel the exact same route, but such that each entity reaches each position with a small delay. We use the term similar-time distance when we allow a bounded time shift when comparing two positions.

3 Experiments

In order to investigate the usability of our definition of a group diagram and the described algorithms for real world data we performed experiments on data of migrating greater white-fronted geese (Anser a. albifrons) with parents and two juveniles. For each animal we had approximately 2000 positions which were collected in half-hourly bursts of 20 GPS positions in 1 Hz resolution. The distance between two entities is computed based on their positions on the earth's surface only. A group diagram 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. Detecting and visualizing split and merge events is an interesting application of the group diagram to help answering questions like: When is the family flying close together, so that it can be represented by only one member and when do we need more representatives? For which distance does the family stay "together" the whole time or a given percentages of the whole observation period? We computed group diagrams based on bounded equal-time and α-similar-time distance as similarity measure for one family (two adults and two juveniles) for distances $d = 3, 5, 10, 20, 40, 80, 160, 320, 640, 1280$ meters and, for each distance, we set the allowed time shift to $\alpha = 0, 10$ seconds. We give a summary of the experiments here, for more details and an evaluation of the experimental computation time see the full version of the paper.

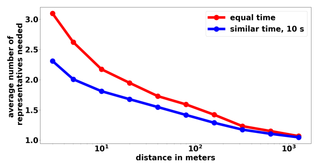


Figure 4 Representability of the family for increasing distance values.

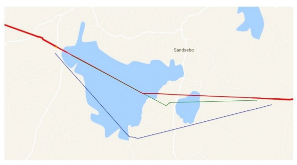


Figure 5 Family members split when flying over a lake and merge after passing the lake.

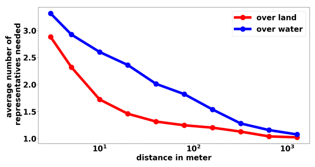


Figure 6 Difference of representability of the family while flying over land and over water.

Number of Representatives for equal- and similar-time distance In Figure 4 the average number of representatives needed to represent the whole family is plotted against the distance thresholds for bounded equal-time distance and similar-time distance as similarity criteria. For small distances (10 m and 20 m) the impact of allowing a time shift of 10 seconds is greater than the impact of doubling the distance. As distance increases it becomes the dominating parameter for the size of the group diagram. The reason for this observation most likely is the formation of the flock while flying. If the entities of the flock are flying within a V-formation or in a line two entities are represented with only one representative even if their distance is greater than the given threshold when we allow a small time shift. The impact of a time shift would be less if the birds were flying next to each other rather than behind each other like in a line or a V-formation.

Migration Over Water and Over Land During the migration one can observe that when the family is flying over surfaces of water they tend to separate more from each other than while flying over solid ground. One example of this phenomenon is shown in Figure 5 for a bounded equal-time distance of 160 meters. Figure 6 shows the difference in the number of representatives needed for flying over solid ground and flying over water. One interesting observation is that the values differ the most between 10 and 100 m.

Acknowledgments. This work was supported by the Deutsche Forschungsgemeinschaft (DFG), project BU 2419/3-1.

References

1 M. Ahmed, S. Karagiorgou, D. Pfoser, and C. Wenk. *Map Construction Algorithms*. Springer, 2015.

2 H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *Int. J. Comput. Geometry Appl.*, 5(1):75–91, 1995.

3 K. Buchin, M. Buchin, J. Gudmundsson, M. Horton, and S. Sijben. Compact flow diagrams for state sequences. In *Proc. Int. Symp. Experimental Algorithms*, pages 89–104, 2016.

4 K. Buchin, M. Buchin, J. Gudmundsson, M. Löffler, and J. Luo. Detecting commuting patterns by clustering subtrajectories. *Int. J. Comput. Geometry Appl.*, 21(3):253–282, 2011.

5 K. Buchin, M. Buchin, M.J. van Kreveld, and J. Luo. Finding long and similar parts of trajectories. *Comput. Geom.*, 44(9):465–476, 2011.

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

7 B. N. Clark, C.J. Colbourn, and D.S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86(1-3):165–177, 1990.