Computing longest duration flocks in trajectory data

Conference Paper · January 2006

DOI: 10.1145/1183471.1183479 · Source: DBLP

2 authors, including:

Joachim Gudmundsson
University of Sydney
175 PUBLICATIONS **2,275** CITATIONS

SEE PROFILE

Computing Longest Duration Flocks in Trajectory Data

Joachim Gudmundsson*
National ICT Australia Ltd
Sydney, Australia.
joachim.gudmundsson@nicta.com.au

Marc van Kreveld†
Institute for Information and Computing Sciences
Utrecht University, Utrecht, The Netherlands.
marc@cs.uu.nl

ABSTRACT

Moving point object data can be analyzed through the discovery of patterns. We consider the computational efficiency of computing two of the most basic spatio-temporal patterns in trajectories, namely flocks and meetings. The patterns are large enough subgroups of the moving point objects that exhibit similar movement and proximity for a certain amount of time. We consider the problem of computing a longest duration flock or meeting. We give several exact and approximation algorithms, and also show that some variants are as hard as MaxClique to compute and approximate.

Categories and Subject Descriptors: F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems

General Terms: Algorithms, Theory.

Keywords: Spatio-temporal patterns, Moving objects, Geometric algorithms, Approximation algorithms.

1. INTRODUCTION

Moving point object data is becoming increasingly more available since the development of GPS and radio transmitters. One of the objectives of spatio-temporal data mining [11, 16, 20] is to analyze such data sets for interesting patterns. For example, a group of caribou with radio collars gives rise to the positions of each caribou in a sequence of time steps. More examples are moose in Sweden (25 animals reported every 30 minutes), leopards in South Africa (32 animals reported daily), mountain goats in USA (32 animals reported every 3 hours), and so on [23]. Analyzing this data gives insight into entity behavior, in particular, migration patterns [18]. The analysis of moving objects also has applications in socio-economic geography [4], transport analysis [19] and in defense and surveillance areas [17].

In general the input is a set P of n moving point objects $p_1, \ldots, p_n$ in the plane whose locations are known at τ consecutive time steps $t_1, \ldots, t_\tau$, that is, the trajectory of each object is a polygonal line that can self-intersect. For brevity, we will call moving point objects *entities* from now on. It is assumed that the velocity of an entity along a line segment of the trajectory is constant.

There are several slightly different definitions of flocks [2, 13, 14]. We will use the following definition, see Fig. 1 (a).

Definition 1. flock(m, k, r)*: Given a set of n trajectories of entities in the plane, where each trajectory consists of τ line segments, a flock in a time interval I, where the duration of I is at least k, consists of at least m entities such that for every point in time within I there is a disk of radius r that contains all the m entities (note that $m \in \mathbb{N}$, $k \in \mathbb{R}$ and $r > 0$ are given constants).*

This definition is almost identical to Definition 1 in [2] with the difference that k is not restricted to be an integer.

We consider two variants of flocks. Either the same m entities stay together during the entire interval (fixed-flock), or the entities in the flock change during the interval (varying-flock). If the entities may change, we require that the disk of radius r changes location in a continuous way. Note that both cases require at least m entities to be within the disc at every moment in I.

A meeting pattern is defined as follows, see Fig. 1 (b):

Definition 2. meet(m, k, r)*: Given a set of n trajectories of entities in the plane, where each trajectory consists of τ line segments, a meeting in a time interval I, where the duration of I is at least k, consists of at least m entities that stay within a stationary disk of radius r during I (note that $m \in \mathbb{N}, k \in \mathbb{R}$ and $r > 0$ are given constants).*

We also consider two variants of meetings: either the same m entities stay together during the entire interval (fixed-meet), or the entities in the meeting region change during the interval (varying-meet). In this paper we consider optimization variants of the flock and meeting problems:

flock$(m, \max, r)$/meet$(m, \max, r)$: compute the longest duration pattern.

We will show that varying-flock, fixed-meet, and varying-meet can be solved in time polynomial in n and τ, whereas fixed-flock is NP-hard. For all four problems, we also present

*NICTA is funded by the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council.
†Supported by the Netherlands Organisation for Scientific Research (NWO) under FOCUS/BRICKS grant number 642.065.503 (GADGET).

ACM-GIS'06, November 10-11, 2006, Arlington, Virginia, USA.

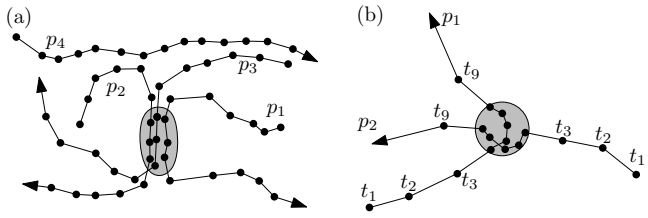

Figure 1: (a) A flock for p_1, p_2, p_3 marked in grey. (b) A meeting for p_1 and p_2 marked in grey.

approximation algorithms by allowing slightly larger radius, i.e., the radius of the reported group may not be bounded by r, but instead it is bounded by cr, where $c > 1$. Consider a longest duration flock F' for the $\mathsf{flock}(m, \max, r)$-problem and let I' denote the time interval during which F' is defined. A flock F during a time interval I is said to be a *c-radius* approximation of the $\mathsf{flock}(m, \max, r)$-problem, if and only if F consists of at least m entities such that for every point in time within I there is a disk of radius cr that contains all the m entities, and I is at least as long as I'. This definition applies to the fixed and varying versions, and is analogous for the meeting patterns.

Approximating the radius of the region or subset size for spatio-temporal data mining was first considered by Gudmundsson et al. [6]. They proposed to approximate the patterns: "Any exact values of m and r hardly have a special significance—20 caribou meeting in a circle with radius 50 meters form as interesting a pattern as 19 caribou meeting in a circle with radius 51 meters." In this paper we only consider approximations of the radius, not of the subset size.

Previous results. From a data mining and database perspective the research has mainly been focusing on modeling, querying and indexing spatio-temporal data, see for example [7, 9, 12, 21]. A common approach in database research is to take an existing spatial query type and then study its generalizations to spatio-temporal data, see for example [8, 15]. Recent advances in data mining have been accomplished by using associated rule mining to distinguish regions with high activity [22], for example, sinks, sources and thoroughfares. A different approach was suggested by Laube and Imfeld [13]: the REMO framework (RElative MOtion) considers similar behavior in groups of entities. They define several spatio-temporal patterns for trajectories, based on similar direction of motion or change of direction. Laube et al. [14] extended the framework by not only including direction of motion, but also location itself. They defined several spatio-temporal patterns, including *flock*, *leadership*, *convergence*, and *encounter*, and gave algorithms to compute them efficiently. Among other results they developed an algorithm for finding the largest flock pattern (maximum number of entities) using the higher-order Voronoi diagram with running time $O(\tau(nm^2 + n \log n))$; they also proved that the detection problem can be answered in $O(\tau(nm + n \log n))$ time. Applying the algorithm by Aronov and Har-Peled [1] to the problem gives a $(1+\varepsilon)$-approximation with expected running time $O(\tau n \log^2 n/\varepsilon^2)$, where the algorithm approximates the flock size. Gudmundsson et al. [6] showed that if the disk is $(1+\varepsilon)$-approximated then the detection problem can be solved in $O(\tau(n \log(1/\varepsilon)/\varepsilon^2 + n \log n))$ time.

However, the algorithms listed above use a different definition of flock than used in this paper. Their definition only considers the entities at one time step, i.e., a set of at least m entities within a circular region of radius r is a flock if they move in the same direction. Benkert et al. [2] argue that this is not sufficient for many applications. Flocks may need many time steps to be properly defined. For example, in some of the aforementioned applications the coordinates of the tracked animals are reported every 30 minutes and a group of animals must stay together for days to form a flock. Benkert et al. [2] recently used a different approach to compute "approximate" flocks. They transform a subpath of length k (assumed to be an integer) of each trajectory into a point in $2k$-dimensional space. Then the problem is reduced to find $2k$-dimensional balls that contain at least m points. They give a $(1+\varepsilon)$-radius approximation to $\mathsf{fixed\text{-}flock}(m, k, r)$ with running time $O(\frac{n\tau k^2}{m\varepsilon^{2k}}(\log n + \varepsilon^{1-2k}))$. Note however that the optimization version of the flock problem is not considered in [2], and the running time of their algorithm is exponential in k.

This paper is organized as follows. In the next section we show that the $\mathsf{fixed\text{-}flock}(m, \max, r)$ problem is as hard as MaxClique to compute and approximate. In Section 3, we study the flock problem further, give a polynomial time, exact algorithm for $\mathsf{varying\text{-}flock}(m, \max, r)$, and approximation algorithms for both variants. In Section 4 we consider the meeting problem and give exact and approximation algorithms for both variants. We conclude with future research. Our algorithmic results are summarized in Table 1.

2. HARDNESS RESULTS

We start with proving two hardness results, which also motivates why we study c-radius approximation algorithms in the following sections. The first concerns maximizing the subset size of the flock, whereas the second concerns maximizing the duration. Both reductions use MaxClique, which is NP-hard [5], and furthermore, cannot be approximated well.

Fact 1. *(Håstad 1999 [10])*
For any constant $\varepsilon > 0$, MaxClique *cannot be approximated in polynomial time within a factor of $n^{1/2-\varepsilon}$ unless $P = NP$, and not within a factor of $n^{1-\varepsilon}$ unless NP=ZPP.*

Theorem 1. *The problem of computing a $(2-\delta)$-radius approximation of $\mathsf{fixed\text{-}flock}(\max, k, r)$, for any $0 < \delta \leq 1$, is NP-hard.*

Proof. The reduction is from MaxClique. Let $G = (V, E)$ be some graph with n vertices, and suppose we wish to determine whether a clique of size m exists.

We construct an instance of the flock problem as follows. Every vertex is represented by an entity. At time step zero, all entities are at the origin of the plane. Assume that $r = 1$; the instance can be scaled to realize any value of r. We define the locations of all entities at all time steps as follows. For $0 < i \leq n$ and time steps $3i - 2$ and $3i$, we let all entities be at $(5i, 0)$. At time step $3i - 1$, the entity of vertex v_i is at $(5i, 2)$, and all entities of vertices that are not connected to v_i in G are at $(5i, -2)$, as shown in Fig. 2. We easily observe that at time $3i - 1$, a circle of radius 1 can contain the i-th entity and all entities whose vertices are connected to v_i in G, or a circle of radius 1 contains all entities except

Subset \ Pattern	Meeting		Flock	
Varying subset	$O(n^4\tau^2 \log n + n^2\tau^3)$	(Exact)	$O(n^3\tau \log n)$	(Exact)
	$O(\frac{n^2\tau}{m\varepsilon^2} \log n \log \tau)$	$((1+\varepsilon)$-apx. $(*))$	$O(\frac{n^2\tau}{\varepsilon^2} \log n)$	$((2+\varepsilon)$-apx.)
	$\Omega(n^2\tau)$	(Any apx.)	$\Omega(n^2\tau)$	(Any apx.)
Fixed subset	$O(n^4\tau^2 \log n + n^2\tau^3)$	(Exact)	NP-hard	(Any $(2-\varepsilon)$-apx.)
	$O(\frac{n^2\tau}{m\varepsilon^2} \log n \log \tau)$	$((1+\varepsilon)$-apx. $(*))$	$O(n^2\tau \log n)$	(2-apx.)
	$\Omega(n^2\tau)$	(Any apx.)	$\Omega(n^2\tau)$	(Any apx.)

Table 1: A summary of the results in this paper. The results marked with a star assume that the longest duration meeting spans a time step. The lower bounds hold in the model of [3].

for v_i. Hence, the question whether a clique of size m exists in G can be answered by answering the question whether a flock of size m exists for the duration of all $k = \tau = 3n$ time steps. This proves NP-completeness of the decision version of the maximum subset fixed flock problem.

The proof holds for any circle radius in $[1, 2)$. Hence, allowing an approximate radius with a factor less than 2 leaves the problem NP-hard. □

The above result can easily be strengthened by noting that an α-approximation of $\mathsf{flock}(\max, k, r)$ also corresponds to an α-approximation of MaxClique, that is, we cannot hope to find a flock of approximately the maximum size efficiently.

Corollary 1. *The problem of computing a $(2-\delta)$-radius approximation of* $\mathsf{fixed\text{-}flock}(\max, k, r)$*, for any constant $0 < \delta < 1$, is at least as hard as approximating* MaxClique.

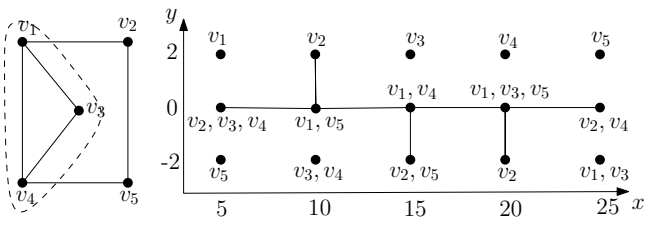

Figure 2: Illustrating the reduction from MaxClique to $\mathsf{fixed\text{-}flock}(\max, k, r)$. The trajectory of v_2 is shown.

3. FLOCK PATTERNS

In the next two sections we will present several algorithms for the flock and meeting problems. In this section we focus on the flock patterns and give three algorithms; a 2-radius approximation algorithm for the $\mathsf{fixed\text{-}flock}(m, \max, r)$ problem, an exact algorithm for the $\mathsf{varying\text{-}flock}(m, \max, r)$ problem and a $(2+\varepsilon)$-radius approximation algorithm for the $\mathsf{varying\text{-}flock}(m, \max, r)$ problem.

3.1 Approximating a fixed flock

We present a 2-radius approximation algorithm for the $\mathsf{fixed\text{-}flock}(m, \max, r)$ problem. That is, if a flock of duration T exists for a radius r disc, then our algorithm finds a flock of duration at least T for a radius $2r$ disc. The algorithm runs in $O(n^2\tau \log n)$ time, where n is the number of entities and τ is the total number of time steps.

Transform all the trajectories into three dimensions where the third dimension is time. For every entity v, consider the region of all points that are within distance $2r$ from v at some moment in time. This region, denoted $\Delta_{2r}(v)$, is represented by a cylindric region, such that every cross-section with a horizontal plane is a disc of radius $2r$, as shown in Fig. 3 (a).

Let v be some entity. Determine for all other entities v_i the time intervals that they are within $\Delta_{2r}(v)$. Since an entity can enter and/or leave the cylinder at most once per time step, the number of intervals is bounded by $O(n\tau)$. From now on we will ignore the entities and only consider the time intervals, see Fig. 3 (b). The problem is now to find a longest interval containing at least m intervals.

The endpoints of the intervals can be sorted in $O(n\tau \log n)$ time, because we merge n sorted sequences of $O(\tau)$ intervals. Go through the $O(n\tau)$ possible start points $t'_1, t'_2, \ldots$ of maximum duration flocks in non-decreasing order. They are the starting times of the intervals. For t'_1, take all $O(n)$ intervals that contain t'_1, sort on the endpoint of these intervals, and store them in an augmented binary search tree where integers with internal nodes represent the number of leaves in the subtree. Then we find the $(m-1)$-th last endpoint and its corresponding time, which is the maximum duration flock for the starting time t'_1 (the m-th entity is the one defining the cylindric region). To process t'_2, we first remove all intervals with endpoint between t'_1 and t'_2, then we add the interval that starts at t'_2 to the augmented binary tree, and again determine the $(m-1)$-th last endpoint. For t'_1 we need $O(n \log n)$ time due to initialization of the tree, and for every subsequent t'_j we only need $O(\log n)$ time if we disregard the removals between t'_{j-1} and t'_j. Throughout the whole process there are only $O(n\tau)$ such deletions, and each deletion can be performed in $O(\log n)$ time. In total there are $O(n\tau)$ operations on the tree, leading to $O(n\tau \log n)$ time to handle $\Delta_{2r}(v)$. We perform these steps for each entity, taking $O(n^2\tau \log n)$ time in total.

It remains to prove that the proposed algorithm gives a 2-radius approximation. Consider a longest duration fixed flock F with entity set f, and let v be an arbitrary entity in f. The algorithm will process v at some point, i.e., it will consider the cylinder $\Delta_{2r}(v)$ and entity v will be at distance at most $2r$ from the $m-1$ other entities in f. Hence, a disc with radius $2r$ centered at v will contain all entities of f during the full duration of F.

Theorem 2. *There is a 2-radius approximation algorithm for the* $\mathsf{fixed\text{-}flock}(m, \max, r)$*-problem that uses $O(n^2\tau \log n)$ time and $O(n\tau)$ space.*

If we treat the entities in random order, we expect to find a flock of duration T already after $O(n/m)$ entities, albeit

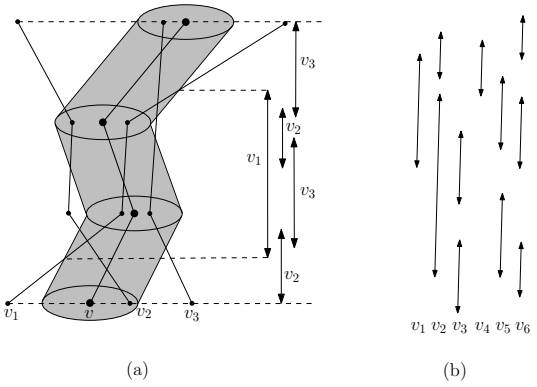

Figure 3: (a) The shaded cylindric region around v is denoted $\Delta_{2r}(v)$. (b) The intervals computed from the intersection of the cylinder.

the algorithm would not know that a flock of duration at least T has been found. Hence, a Monte Carlo algorithm for longest duration flock takes $O(\frac{n^2\tau}{m}\log n)$ time.

Remark. It is likely that every c-radius approximation algorithm requires $\Omega(n^2\tau)$ time. Consider the flocking problem on the real line instead of the plane, and suppose we wish to determine a flock of size at least 3 for radius 0. We can transform the n moving entities between time steps t_i and t_{i+1} to n lines in the plane, where the second dimension represents time. A flock of size at least 3 exists if and only if at least 3 lines pass through a single point between t_i and t_{i+1}. For this problem, a lower bound of $\Omega(n^2)$ is known, albeit in a weak model of computation [3].

Using the same argumentation one can argue that the running time for any c-radius approximation algorithm for each of the four problems considered in this paper (varying/fixed-flock/meet problem) is bounded by $\Omega(n^2\tau)$.

3.2 Longest duration varying flock

In this section we consider the varying-flock$(m, \max, r)$ problem, in other words, we wish to determine the longest time interval for a continuously moving circle of radius r such that it contains at least m entities throughout the whole interval. Recall that the set of entities inside the circle need not be the same throughout the flock duration.

Let F be a longest duration varying flock, and assume the time interval of F is I. The set of entities of F at time $t \in I$ is denoted $f(t)$.

PROPERTY 1. *At every fixed moment t in I there exists a disc D of radius r that contains $f(t)$, such that at least two entities of $f(t)$ lie on the perimeter of D.*

Assume that at some point t the set of entities in the flock changes. In general this change can happen at any time during a time interval, however, if we assume that the number of entities is always maximized then an entity will join as early as possible, an entity will leave as late as possible, and an exchange may either occur as early as possible or as late as possible. The following property then holds.

PROPERTY 2. *Consider a moment $t \in I$ when the set of entities in the flock changes. There is a disc D of radius r that contains $f(t)$ with either three entities on the boundary, or two points on the boundary at distance $2r$.*

Next we design an algorithm making use of the above properties. Let us first study the problem between two consecutive time steps, so that each entity moves with constant speed along a line. The final running time is obtained by multiplying the running time needed to handle one time step with the total number of time steps.

Imagine any disc movement through time for a varying flock, the disc moves continuously with two entities on the boundary, until a third entity lies on the disc as well, according to Properties 1 and 2. It may come from the inside and go out, or it may come from the outside and go in. At this moment, there is a choice which two out of the three entities will determine the motion of the disc. An example is given in Fig. 4 (a) where p_1 and p_2 initially define a disc, and at some point in time p_2 leaves the flock and the two entities p_1 and p_3 define the disc. Note that if the third point on the circle will go outside, it may be that the flock stops because there is no longer a disc with at least m points inside.

Between two consecutive time steps, the entities can be viewed as straight lines in 3-space. Every triple of lines defines a constant number of moments where all three lie on the boundary of a circle of radius r. Furthermore, every pair of lines has a constant number of moments where they are at distance exactly $2r$. These two types of event are the only ones where a flock can start, end, or exchange a point that defines the circle. From the above discussion the following observation trivially follows.

OBSERVATION 1. *The number of events between two consecutive time steps is $O(n^3)$.*

Next we show how to handle the events efficiently. For each pair of lines in space ℓ and ℓ', and for each fixed moment t, there are (at most) two discs of radius r with ℓ and ℓ' on its boundary. As a first step, compute all time intervals where ℓ and ℓ' define a disc of radius r with at least $m-2$ other lines inside, and no third line on the boundary. These open intervals can be computed by tracing each disc through time and computing all intervals for which the disc contains at least $m-2$ other entities. This can be done in $O(n\log n)$ time per disc. At the start point or end point of each time interval for which ℓ and ℓ' lie on the boundary of a disc, there is a third line ℓ'' on the boundary, or ℓ and ℓ' are at distance exactly $2r$. In the first case, there are also intervals for the pairs ℓ, ℓ'' and ℓ', ℓ'' that have the same disc with ℓ, ℓ' and ℓ'' on the boundary as start point or end point. When the intervals for two lines are computed one can also compute all the events for which three lines lie on a circle of radius r and contain at least $m-3$ other lines inside, thus the total time required to compute all the events is $O(n^3\tau\log n)$.

We define a directed acyclic graph $G = (V, E)$ where each node in V represents a disc of radius r through three lines at a time instance, or through two lines and they are at distance $2r$. Two nodes are connected by an edge in E if these two nodes represent discs that are the start point and end point of a time interval for a pair. We direct the edge from earlier to later in time, and assign a weight that is its duration. Then we compute a maximum cost directed path in G by a bottom-to-top traversal. Starting at all leafs we propagate the cost of the longest path up the graph, thus each node v will store the value of the maximum cost path

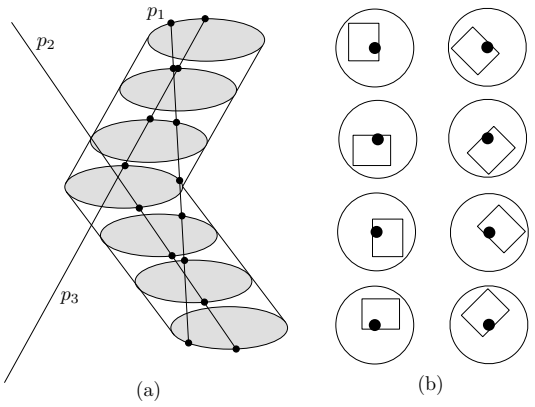

Figure 4: (a) The entities p_1 and p_2 initially define a disc, and at some point in time p_2 leaves the flock and the two entities p_1 and p_3 define the disc. (b) Illustrating the placement of the rectangles with respect to the entities.

to any of its descendants. We claim that this path defines a longest duration flock. The claim follows since the graph models all the events and all the possible outcomes of an event, thus Properties 1 and 2 guarantee that the optimal solution is the maximum weight path in G.

THEOREM 3. varying-flock$(m, \max, r)$ *can be computed in time* $O(n^3\tau \log n)$.

3.3 Approximating a varying flock

In this section we will present a c-radius approximation algorithm for the varying-flock$(m, \max, r)$-problem, for any constant $c > 2$. As in the exact algorithm, the idea is to build a graph on possible movements of the disc, based on the geometry. A path in the graph will correspond to a continuous movement of a horizontal disc of radius cr through time with m entities inside the disc at every moment.

For each pair of entities we will generate two types of events: type 1 and type 2 events. Consider an entity v. A type 1 event is generated at every moment when another entity, say v', is at distance exactly cr from v; the event corresponds to the disc of radius cr with center at v and with v' on its boundary. A type 2 event is generated at every moment when another entity is at distance exactly $2r$ from v. There will be $O(n\tau)$ events per entity, so $O(n^2\tau)$ events overall.

We define a graph $G = (V, E)$, where each node in V represents an event. We will add two sets of edges to the graph. Consider an entity v and every two consecutive events generated by v. If the cylindric region $\Delta_{cr}(v)$ (Figure 3 (a)) contains at least m entities between the two events then an edge is added between the two corresponding nodes of V. The weight of the edge is the time span between the two events. The edges can be computed in time $O(n^2\tau \log n)$, as described in Section 3.1. These edges represent a path within the cylindric region of one entity only, so we must consider what other edges are needed to represent the situation that a flock may consist of a changing subset of m entities. The second set of edges is to guarantee that a disc centered at the one entity can be moved horizontally to a disc centered at the other entity while keeping at least m entities inside. If that is the case, then we add an edge between the two nodes in G. More precisely, for any two type 2 event nodes ν_i and ν_j in V of entities v_i and v_j, we add an edge to G from ν_i to ν_j if (i) the corresponding events occur at the same time t; (ii) v_i and v_j are at distance exactly $2r$ at time t, and (iii) there is a disc of radius r in the intersection of $\Delta_{cr}(v_i)$ and $\Delta_{cr}(v_j)$ with v_i on its boundary that contains at least m entities at time t.

A slight complication is that the second set of edges may generate cycles in G; an edge from ν_i to ν_j and an edge from ν_j to ν_i. A simple graph augmentation can easily fix the problem: we make two nodes ν_i and ν_i' for v_i with a directed, zero-duration edge from ν_i to ν_i'. We do the same for v_j. Then we make a zero-duration edge from ν_i to ν_j', and another zero-duration edge from ν_j to ν_i'. The non-zero duration edges from ν_i are moved to originate from ν_i'.

THEOREM 4. *If the longest path in G has weight W then*
1. there is a c-radius approximation of meet$(m, \max, r)$ *with duration at least W, and*
2. meet$(m, \max, r)$ *has duration at most W.*

Summarizing our results, the total number of events is $O(n^2\tau)$ and the total number of edges is bounded by $O(n^2\tau)$ as well. Furthermore, the set of edges can be computed in $O(\frac{n^2\tau}{(c-2)^2} \log n)$ time, as will be shown in Section 3.3.1. As a result the graph is constructed in $O(\frac{n^2\tau}{(c-2)^2} \log n)$ time, and if the longest duration flock has duration I, then G contains a path of duration at least I. As in the previous section, we compute a maximum cost directed path in G by a bottom-to-top traversal. Starting at all leafs we propagate the cost of the longest path up the graph, thus each node v will store the value of the maximum cost path to any of its descendants. We obtain:

THEOREM 5. *A c-radius approximation of* varying-flock $(m, \max, r)$ *can be computed in time* $O(\frac{n^2\tau}{(c-2)^2} \log n)$*, for any constant $c > 2$.*

3.3.1 Computing valid transitions

In this section we consider the problem of deciding if the intersection of two cylinders, $\Delta_{cr}(v_i)$ and $\Delta_{cr}(v_j)$, at time t contains a disc of radius r with v_i on its boundary and at least $m-1$ other entities inside. This step will be too time consuming if exact queries would be performed, instead we will approximate the number of entities within a radius-r disc. As a consequence some extra edges will be added even if the above property is not fulfilled. However, an edge will only be added if the number of entities within the intersection of $\Delta_{cr}(v_i)$ and $\Delta_{cr}(v_j)$ contains at least $m-2$ points, which is a property that suffices to prove Theorem 4.

The idea is to define a constant number of (overlapping) skew boxes for each entity. Each skew box can be seen as a rectangle at a fixed moment in time. The position of the rectangle will be fixed with respect to the entity, as illustrated in Fig. 4 (b). Produce a set of skew boxes for each entity v, such that at every moment and for every radius-r disc that has v_i on its boundary the disc is contained in at least one of the skewed boxes.

One way to achieve the above properties is to place a constant number of rectangles around each entity that then are traced through time. The number of rectangles, denoted κ, can be bounded by $O(1/(c-2)^2)$ which can be derived

using standard trigonometric calculations. The placement is illustrated in Fig. 4 (b).

For every entity v, consider the κ boxes obtained by moving the rectangles along the trajectory described by v; in total there are $O(n\tau\kappa)$ boxes. For every box, trace the rectangle through time and maintain the entities that enter and leave. Since there are $O(\tau)$ events per entity, a box can be processed in $O(n\tau \log n)$ time, so $O(n^2\tau\kappa \log n)$ time overall. In this way, we can compute the time intervals for each box for which there are at least m entities inside.

Given two entities v_i and v_j at distance exactly $2r$ at time t, let $C(v_i)$ and $C(v_j)$ be the two discs of radius cr with center at v_i and v_j respectively. One can decide in $O(\kappa \log n)$ time if there is a rectangle in the intersection of $C(v_i)$ and $C(v_j)$ that contains at least m points. Note that the above construction guarantees that the disc of radius r with v_i and v_j on its boundary will lie entirely within the rectangle. Computing all the valid transitions can in this way be done in $O(n^2\tau\kappa \log n)$ time.

4. MEETING PATTERNS

In this section we solve the problem of computing longest duration meeting patterns. Unlike the corresponding flock pattern this problem is not NP-complete and below we will give a polynomial time algorithm for fixed-meet$(m, \max, r)$, followed by a faster $(1+\varepsilon)$-radius approximation algorithm.

As in the previous section we consider the n trajectories in three dimensions, where the third dimension is time. The longest duration meeting with radius r is represented by a vertical column of maximum height and radius r, where m entities are inside throughout the full height of the column. For the fixed subset meet pattern these must be the same m entities, for the varying subset meet they may be different. In both versions, the bottom and top disks of the column represent the start and end time of the longest meeting.

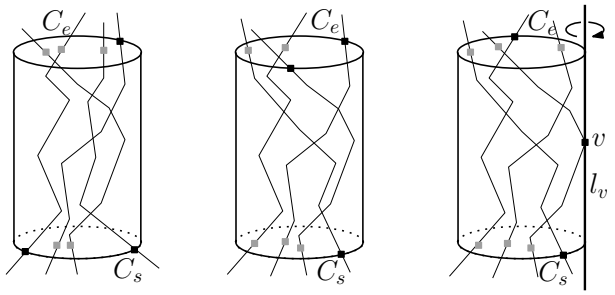

Figure 5: The three cases of a maximum height column. Grey squares show where a trajectory passes through the top or bottom disk, and black squares show intersection points or vertices that determine the column. In the rightmost case the vertical line around which is swept is also shown.

A maximum height, fixed radius vertical column can be described by four parameters. We need up to four entities to fix such a shape. Clearly, one entity must have its trajectory pass through the bounding circle C_s at the start time and one entity must have its trajectory pass through the bounding circle C_e of the end time. Furthermore, one of the following situations must also arise in the maximum height column (see Figure 5):

1. One more trajectory passes through C_s.
2. One more trajectory passes through C_e.
3. A vertex of a trajectory lies on the boundary of the column between C_s and C_e; this may be a vertex of the trajectory that passes through C_s or C_e.

The cases do not capture exactly when a maximum height column arises, but they contain all possibilities. Hence, if we examine all these cases, we will not miss the maximum height column. Cases 1 and 2 are symmetric so we will describe only one of them. We start with the description of the fixed subset meeting pattern, then we briefly describe what must be changed to adapt the algorithm for the varying subset meeting pattern.

4.1 Longest duration fixed meeting

Assuming case 1, we make all possible choices of the two edges that determine C_s, and sweep a column of radius r in contact with these edges. More precisely, if the two edges are in the time interval $[t_i, t_{i+1}]$, we sweep through this time interval so that the two edges give two points which determine two possible vertical columns. We need to determine the maximum height column that starts at the time moment that we are sweeping. In total, we must do $O(n^2\tau)$ such sweeps.

Consider any one such sweep. We will maintain information on all intersections of trajectories with the vertical half-column above the lower disk whose boundary is in contact with the two chosen edges. In particular, for the active subset of trajectories (which intersect the lower disk), we maintain the order in which they leave the half-column. Consequently, we always know when the m-th last trajectory leaves the half-column, which determines the maximum height of the column for the current lower disk location.

When the half-column moves "up and sideways" to keep the contacts, the following events occur. (i) A vertex of some trajectory leaves or enters the half-column. (ii) The lower disk intersects an edge (a trajectory enters or leaves the initial subset). (iii) The order in which trajectories leave the half-column changes (swap in the time-coordinate of two intersections of trajectories with the half-column). There are $O(n\tau)$ events of type (i), $O(n)$ events of type (ii), and $O(n^2\tau)$ events of type (iii). Events of type (i) can easily be handled in $O(\tau + \log n)$ time by checking the $O(\tau)$ vertices and edges of the trajectory against the current half-column. Possibly, we must update leaving order of the trajectories by one deletion and one insertion. Events of type (ii) can also be handled in $O(\tau + \log n)$ time in the same way. Events of type (iii) take $O(\log n)$ time. Events of type (i) and (ii) are precomputed, whereas type (iii) events are computed on the fly. We conclude that the whole sweep process takes $O(n^2\tau \log n)$ time. Hence, all sweeps together take $O(n^4\tau^2 \log n)$ time.

Case 3 is treated as follows. We choose any vertex v of a trajectory, take the vertical line l_v through it, and rotationally sweep a column of radius r that contains l_v. Let D_v be the horizontal disk in the column with v on its boundary. We will maintain for all trajectories that intersect D_v the time interval that they are inside the column, but only the interval that contains the time of vertex v. Since the start and end times of these intervals change continuously when the column rotates, we only maintain the order of entry of the trajectories in the column, and the order of leaving the

column. For any location of the column in the sweep, the exact times of entry and leaving can be computed easily.

The following events occur. (i) A vertex of some trajectory leaves or enters the column. (ii) D_v intersects an edge (a trajectory enters or leaves the subset at the time of v). (iii) The order in which trajectories enter or leave the column changes (swap in the time-coordinate of two intersections of trajectories with the column). There are $O(n\tau)$ events of type (i) and they can be handled in $O(\tau + \log n)$ time. There are $O(n)$ events of type (ii) and they can be handled in $O(\tau + \log n)$ time. There are $O(n^2\tau)$ events of type (iii) and they can be handled in $O(\log n)$ time. Events of type (i) and (ii) are precomputed, whereas type (iii) events are computed on the fly. After restoring the orders at an event, we check in $O(n)$ time by a scan through the two ordered sets what the longest meet is for this location of the column. Actually, we compute this for a location and also for angular intervals between two sweep events, where the structure does not change. Multiplying and adding gives an $O(n^3\tau + n\tau^2)$ time bound for a single sweep in case 3. We need to do $O(n\tau)$ such sweeps.

THEOREM 6. fixed-meet(m, max, r) *can be computed in* $O(n^4\tau^2 \log n + n^2\tau^3)$ *time.*

4.2 Longest duration varying meeting

To solve the varying subset longest meeting problem, we use the same two sweeps as in the fixed subset longest meeting problem. The main difference is that we must maintain all connected components of trajectories inside the column or half-column. Consequently, for the half-column sweep of case 1, type (ii) events are replaced by all events where an edge intersects the boundary of the half-column; there can be $O(n\tau)$ of such events. We maintain all $O(n\tau)$ intervals of trajectories inside the half-column and the order of their endpoints in an augmented segment tree; augmentation stores the minimum coverage of any point (in time) in the subtree. We do not need to know explicitly which intervals are stored with nodes, only how many. Therefore, we can use a segment tree for stabbing counting queries. The running time for a sweep of this type is $O(n^2\tau \log n\tau)$.

For the rotating column sweeps of case 3, we also adapt the type (ii) events and will have $O(n\tau)$ of them. It is easy to again attain the $O(n^3\tau+n\tau^2)$ time bound for a single sweep, since we only need to know the number of trajectories inside the column at each point in time.

THEOREM 7. varying-meet(m, max, r) *can be computed in* $O(n^4\tau^2 \log n + n^2\tau^3)$ *time.*

We could replace the factor $\log n\tau$ by $\log n$, because either these bounds are asymptotically the same, or the second term $n^2\tau^3$ dominates the running time anyway.

4.3 Approximating a fixed meeting

We give a $(1+\varepsilon)$-radius approximation for the fixed-meet $(m, \max, r)$ problem. There are two cases to consider: either the meeting spans a time step, or the meeting lies within one time step. We start with the first case.

Case 1. Let F be a longest duration fixed meeting and let f be the entities of F. Assume that F spans some time step t. At some time step t, let C_{opt} be a disc of radius r that contains f; disc C_{opt} has an unknown location and occurs at an unknown time step.

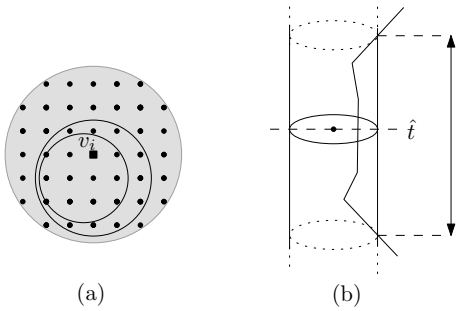

Figure 6: (a) One of the generated discs of radius $(1+\varepsilon)\cdot r$ will contain C_{opt}. (b) A vertical column from the disc at time $\hat{t}$ both upwards and downwards.

We will first try a time step $\hat{t} = \tau/2$. We determine a set of discs of radius $(1+\varepsilon)r$ such that at least one of them will contain the optimal disc C_{opt}, if F occurs at $\hat{t}$. For each entity v_i, place $O(1/\varepsilon^2)$ points inside a circle of radius $2r$ centered at v_i in a regular grid configuration with spacing $\varepsilon \cdot r/4$. These points will be centers of discs with radius $(1+\varepsilon)\cdot r$ which we will test further. Note that if C_{opt} contains v_i, then one of the generated $O(1/\varepsilon^2)$ discs of radius $(1+\varepsilon)\cdot r$ will contain C_{opt} completely, as illustrated in Fig. 6 (a).

For all n entities we generate $O(n/\varepsilon^2)$ discs to be tested further. We may generate the same disc multiple times if its center lies closer than $2r$ from multiple entities, and we use the same base grid to choose center points from. However, we only need to test each disc once. Furthermore, we are only interested in discs that contain at least m entities thus the same disc will in this case be generated m times but only tested once. A simple counting argument now shows that only $O(n/(m\varepsilon^2))$ distinct discs contain at least m points.

To test a disc, we consider the time dimension again, and erect a vertical column from the disc both upwards and downwards, see Fig. 6 (b). We trace the trajectories of all entities that are in the disc at time $\hat{t}$ and test when they first leave the column, both in upward and downward direction. By examining the two ordered sequences of departure times we can find the longest duration meeting of the radius $(1+\varepsilon)\cdot r$ disc, which is assumed to span time $\hat{t}$. Testing a disc takes $O(n\tau \log n)$ time.

Next we go into recursion on the time spans before and after $\hat{t}$; the relevant trajectories contain $\tau/2$ time steps in each half. The recursion bottoms out when all time steps have been considered. If a longest duration meeting spans at least one time step then we are certain to have found a radius $(1+\varepsilon)\cdot r$ meeting of duration at least as long as the longest radius-r meeting in $O(\frac{n^2\tau}{m\varepsilon^2} \log n \log \tau)$ time.

THEOREM 8. *For any given constant $\varepsilon > 0$, if a longest duration meeting spans a time step, then a $(1+\varepsilon)$-radius approximation of* fixed-meet$(m, \max, r)$ *can be computed in time* $O(\frac{n^2\tau}{m\varepsilon^2} \log n \log \tau)$.

Case 2. To approximate the longest duration meeting between two time steps t_i and t_{i+1} when the duration can be arbitrarily short, we need to use an approach that is less

efficient. At time t_i, compute the $O(n/\varepsilon^2)$ disks of radius $(1+\varepsilon)\cdot r$ as above. However, this time we do not remove the ones that contain less than m entities, nor do we remove duplicates. Consider one disk, corresponding to an entity v_j. We sweep a vertical column abutted at time t_{i+1} upwards in the same direction as v_j, and maintain when the other entities are inside. The sweep is similar to the one for exact meetings, but simpler. All $O(n^2)$ events can be handled in $O(\log n)$ time because we only consider subtrajectories between two time steps. Hence, all $O(n\tau/\varepsilon^2)$ sweeps take $O(\frac{n^3\tau}{\varepsilon^2}\log n)$ time in total.

THEOREM 9. *For any given constant $\varepsilon > 0$, a $(1+\varepsilon)$-radius approximation of* fixed-meet$(m, \max, r)$ *can be computed in time $O(\frac{n^2\tau}{m\varepsilon^2}\log n \log\tau + \frac{n^3\tau}{\varepsilon^2}\log n)$.*

4.4 Approximating a varying meeting

The algorithm is almost identical to case 1 of the above algorithm that computes a $(1+\varepsilon)$-radius approximation of fixed-meet$(m, \max, r)$, given that the meeting spans a time step. We perform divide-and-conquer on τ as before, and test the same $O(n/(m\varepsilon^2))$ vertical columns.

To test a column at time $\hat{t}$, perform two sweeps of a disk inside the column starting at $\hat{t}$ to the top and to the bottom. We determine for all entities the time intervals that they are within the sweeping disk. Since an entity can enter and/or leave the column at most once per time step, the number of intervals is bounded by $O(n\tau)$. The problem is now to find a longest interval containing at least m intervals, which is done before and after $\hat{t}$ independently.

Next we go into recursion on the time spans before and after $\hat{t}$; the relevant trajectories contain $\tau/2$ time steps in each half. If a longest duration meeting spans at least one time step then we are certain to have found a radius $(1+\varepsilon)\cdot r$ meeting of duration at least as long as the longest radius-r meeting; and its running time so far is $O(\frac{n^2\tau}{m\varepsilon^2}\log n\log\tau)$.

To detect the pattern within two time steps we have to spend extra time and sweep abutted columns, like before. The running time is the same as for the fixed-meet problem.

5. CONCLUSIONS

In this paper we considered the problem of computing the longest duration flock, or meeting, given a set of n moving points in the plane. We gave several exact algorithms and approximation algorithms, and we also proved that the problem of computing a $(2-\varepsilon)$-radius approximate longest duration flock is as hard as MAXCLIQUE.

There are many open problems in this area, for example, define and design algorithms for more complex patterns and develop algorithms that can handle spatio-temporal data with errors or with missing information.

6. REFERENCES

[1] B. Aronov and S. Har-Peled. On approximating the depth and related problems. In Proc. 16th ACM-SIAM Symposium on Discrete Algorithms, pages 886–894, 2005.

[2] M. Benkert, J. Gudmundsson, F. Huebner and T. Wolle. Reporting flock patterns. In Proc. of the 14th European Symposium on Algorithms, 2006.

[3] J. Erickson and R. Seidel. Better lower bounds on detecting affine and spherical degeneracies. Discrete & Computational Geometry 13:41–57, 1995.

[4] A. U. Frank and J. F. Raper and J.-P. Cheylan (Eds.). Life and motion of spatial socio-economic units. Taylor & Francis, 2001.

[5] M. R. Garey and D. S. Johnson. Computers and intractability. W. H. Freeman, 1979.

[6] J. Gudmundsson, M. van Kreveld and B. Speckmann. Efficient Detection of Patterns in 2D Trajectories of Moving Points. To appear in GeoInformatica, 2006.

[7] R. H. Güting and M. Schneider. Moving objects databases. Morgan Kaufmann Publishers, 2005

[8] M. Hadjieleftheriou, G. Kollios, P. Bakalov and V. J. Tsotras. Complex Spatio-Temporal Pattern Queries. In Proc. 31st International Conf. on Very Large Data Bases, pages 877–888, 2005.

[9] K. Hornsby and M. Egenhofer. Modeling moving objects over multiple granularities. Annals of Mathematics and Artificial Intelligence, 36(1-2):177–194, 2002.

[10] J. Håstad. Clique is hard to approximate within $n^{1-\varepsilon}$. ACTA Mathematica, 182:105–142, 1999.

[11] P. Kalnis, N. Mamoulis and S. Bakiras. On discovering moving clusters in spatio-temporal data. In Proc. 9th International Symposium on Spatial and Temporal Databases, pages 364–381, 2005.

[12] G. Kollios, S. Sclaroff and M. Betke. Motion mining: discovering spatio-temporal patterns in databases of human motion. In ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, 2001.

[13] P. Laube and S. Imfeld. Analyzing relative motion within groups of trackable moving point objects. In Proc. 2nd International Conference on Geographic Information Science, pages 132–144, 2002.

[14] P. Laube, M. van Kreveld and S. Imfeld. Finding REMO – detecting relative motion patterns in geospatial lifelines. In Proc. 11th International Symp. on Spatial Data Handling, pages 201–214, 2004.

[15] N. Mamoulis, H. Cao, G. Kollios, M. Hadjieleftheriou, Y. Tao and D. W. Cheung Mining, Indexing, and Querying Historical Spatiotemporal Data. In Proc. 10th International Conf. on Knowledge Discovery and Data Mining, 2004.

[16] H. J. Miller and J. Han (Eds.) Geographic data mining and knowledge discovery, Taylor & Francis, 2001.

[17] R. T. Ng. Detecting outliers from large datasets. In H. J. Miller and J. Han, editors, Geographic data mining and knowledge discovery, pages 218-235. Taylor & Francis, 2001.

[18] Porcupine Caribou Herd Satellite Collar project. `http://www.taiga.net/satellite/`.

[19] Y. Qu, C. Wang and X. S. Wang. Supporting fast search in time series for movement patterns in multiple scales. In Proc. 7th International ACM Conference on Information and Knowledge Management, 1998.

[20] J. Roddick, K. Hornsby and M. Spiliopoulou. An updated bibliography of temporal, spatial, and spatio-temporal data mining research. In Proc. 1st Int. Workshop on Temporal, Spatial, and Spatio-Temporal Data Mining, pages 147–163, 2001.

[21] N. Sumpter and A. J. Bulpitt. Learning spatio-temporal patterns for predicting object behaviour. Image Vision and Computing, 18(9):697–704, 2000.

[22] F. Verhein and S. Chawla. Mining spatio-temporal association rules, sources, sinks, stationary regions and thoroughfares in object mobility databases. In Proc. of the 11th International Conference on Database Systems for Advanced Applications, 2006.

[23] Wildlife tracking projects with GPS GSM collars. `http://www.environmental-studies.de`.