Reporting Leaders and Followers Among Trajectories of Moving Point Objects

Mattias Andersson[1], Joachim Gudmundsson[2], Patrick Laube[3,⋆], and Thomas Wolle[2]

[1] Department of Computer Science, Lund University, Lund, Sweden
mattias@cs.lth.se
[2] NICTA⋆⋆ Sydney, Locked Bag 9013, Alexandria NSW 1435, Australia
{joachim.gudmundsson, thomas.wolle}@nicta.com.au
[3] Department of Geomatics, The University of Melbourne, Victoria 3010, Australia
plaube@unimelb.edu.au

Abstract. Widespread availability of location aware devices (such as GPS receivers) promotes capture of detailed movement trajectories of people, animals, vehicles and other moving objects, opening new options for a better understanding of the processes involved. In this paper we investigate spatio-temporal movement patterns in large tracking data sets. We present a natural definition of the pattern 'one object is leading others', which is based on behavioural patterns discussed in the behavioural ecology literature. Such leadership patterns can be characterised by a minimum time length for which they have to exist and by a minimum number of entities involved in the pattern. Furthermore, we distinguish two models (discrete and continuous) of the time axis for which patterns can start and end. For all variants of these leadership patterns, we describe algorithms for their detection, given the trajectories of a group of moving entities. A theoretical analysis as well as experiments show that these algorithms efficiently report leadership patterns.

Keywords: moving point objects, trajectories, movement patterns, leadership, spatio-temporal data structures, computational geometry

1 Introduction

Movement is the spatio-temporal process par excellence. Technological advances of location-aware devices, surveillance systems and electronic transaction networks produce more and more opportunities to trace moving individuals. Consequently, an eclectic set of disciplines including geography [17], data base research [23], animal behaviour research [26], surveillance and security analysis [46, 48, 58], transport analysis [30, 34], and market research [49] shows an increasing interest in movement patterns of various entities moving in various spaces over various times scales.

At the same time traditional geographic analysis suffers from the legacy of cartography's static perception of the world and is thus generally not suited for the analysis of individual movement trajectories [8, 51], sometimes referred to as geospatial lifelines especially in a GIScience context [43]. Many authors have therefore recently proposed to use geographical (and thus) spatio-temporal data mining as a promising alternative to overcome this methodological shortcoming [14, 44].

As can be seen from the pattern terminology, the present paper is largely inspired by movement patterns observed in gregarious animals, such as flocking sheep or schooling fish. It follows a strategy to link the proposed patterns as close as possible to observable patterns. The proposed pattern definitions are based on behavioural patterns discussed in the behavioural ecology literature and used for the modelling of realistic movement patterns of agent-based virtual life forms [27, 35].

This paper addresses the movement pattern of one object leading others. The paper therefore defines the movement pattern 'leadership' and subsequently presents algorithms to detect such

⋆ Partially supported by ARC Discovery grant DPDP0662906.
⋆⋆ National ICT Australia is funded through the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council.

patterns. Leadership, as defined in this paper, bases on the geometrical relation of one individual moving in front of its followers. The algorithms presented for an efficient detection of 'leadership' make use of a set of auxiliary data structures, specifically developed for capturing those spatio-temporal relations amongst moving objects that constitute leadership.

Even though the leadership pattern in this paper is motivated and investigated with respect to animal behaviour research, its definition is held generic and is thus applicable to arbitrary types of entities moving in a 2D-space. In general the input is a set of n moving point objects $e_1, ..., e_n$ whose locations are known at τ consecutive time-steps $t_1, ..., t_\tau$, 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. We assume that the movement of an entity from its position at a time-step t_j to its position at the next time-step t_{j+1} is described by the straight-line segment between the two coordinates, and that the entity moves along the segment with constant velocity.

The paper is organised as follows. Section 2 links previous research on movement patterns similar to our leadership with the latest related research. Section 3 defines our notion of leadership and features definitions and preliminaries. In Section 4 and 5, we present algorithms for the detection of leadership. Then, in Section 6 we present experimental results and discuss their implications in Section 7. We conclude the paper with final remarks and an outlook on future work in Section 8.

2 Related work

2.1 Inspiring Animals

Animals interact socially to gain from coordination of their behaviour [9, 33]. Rands et al. [50] illustrated the spontaneous emergence of leaders and followers using a simulation model reproducing the decision process of a pair of foraging animals, balancing their energetic states. The idea and the term of leadership have been used in several different contexts in the field of animal behaviour research, see Dumont et al. [12] for an overview. In general, one can distinguish two different readings:

1. (i) 'the event or process of one entity initiating a group movement (e.g. [7, 12, 40, 47])' Leading in this sense is an active behaviour, referring to individuals that consistently initiate displacement of the group they belong to. For example, Dumont et al. [12] found that in a group of 15 grazing heifers the same individual was reported to lead the group to new feeding places in 48% of all group movements. Similar leadership behaviour has also been studied in gray wolves (*Canis lupus*) [47].
2. (ii) 'the event or process of one entity in front, leading a group movement (e.g. [7, 22])' Leading in this sense involves the notion of a leader moving in front of followers. Gueron and Levin model the spatial constellation 'in front of' as a function of the relative position with respect to the averaged position of its neighbours within a given range. Even though it has been found for grazing animals that leaders may guide a group being in front or chasing from behind, animals in front are considered to be more relevant to determine where the group will graze [12].

The use of the geometrical arrangement of moving entities has furthermore a long tradition for realistically modelling group behaviour, be it in animal behaviour science [22] or in the animation industry [52]. Most prominent is the flocking model implemented in NetLogo [59, 62], which mimics the flocking of birds [61]. The moving agents dynamically coordinate their movement based on rules on alignment (turning in order to adopt direction of nearby agents), separation (turning to avoid getting to close to nearby agents) and cohesion (move towards other nearby agents). This model explicitly excludes the idea of an individual leading the others, but involves identical agents, each following the same set of rules. The basic model includes a maximal distance of vision r and 360 degree field of view. However, it is also possible in NetLogo to specify a cone of vision, a most interesting concept with respect to the investigation of further structure in flocking entities that can, for example, be seen in V-shaped flocks as with migrating geese. Such front priority is also

often used for agent-based models of schooling fish, where only individuals in front are candidates as interacting neighbours [27]. Inada's and Kawachi's model uses a wide-angled cone of perception, directed in movement direction and thus omitting a blind region behind the fish (Figure 1, Figure 2 in [27]). Jadbabaie et al. give a theoretical explanation for the spontaneous coordination of agents despite the absence of a centralised coordination and just following a simple nearest neighbour rule [29]. However, in an extension they also investigate the influence of a leader in their system.
▶ All such research integrating biology with information science and computer science points out the potential of a systematic investigation of geometric relations of moving animals for analysing, modelling and simulating movement processes. Above all, animal movement provides a set of very convincing metaphors for more generic movement patterns, as shall be exploited with the pattern 'leadership' in this paper.

2.2 Limiting Databases

There is ample research on moving object databases (MOD) [25, 56, 63, 64]. Whereas most database research on MOD focuses on data structures, indexing and efficient querying techniques for moving objects [1, 16, 23, 24], only recently the potential of data mining for movement patterns has been acknowledged [31, 32, 53]. For example, Du Mouza and Rigaux propose mobility patterns that describe sequences of moves in a discrete 2D-space [11].

In a GIScience context, activity related movement patterns have been researched, often with respect to improving location-based services (LBS). Dykes and Mountain search episodes expressing distinctive characteristics of movement, including absolute speed, direction, sinuosity and measurements of their variations [13]. Smyth presents a data mining algorithm that assigns predefined activities to segments of trajectories by analysing some measurable motion descriptors, such as speed, heading and acceleration [57].

A common approach in database research is to take an existing spatial query type and then study its generalisations to spatio-temporal data. An example of this is the recent work on continuous k-nearest neighbour querying over mobile data [45, 65]. The focus within data mining research is to design techniques to discover new patterns in large repositories of spatio-temporal data. For example, Mamoulis et al. [42] mine periodic patterns moving between objects and Ishikawa et al. [28] mine spatio-temporal patterns in the form of Markov transition probabilities. More recently Verhein and Chawla [60] used association rule mining for patterns such as, sinks, sources, stationary regions and thoroughfares.

Spatio-temporal proximity of entities is a reasonable first premise for many situations that assume interactions between individuals. One obvious analytical toolset to uncover proximity patterns in individual trajectories is clustering. Even though the spatio-temporal nature of movement data adds additional complexity to clustering procedures, there have been some successful approaches for clustering trajectories [10, 41, 55]. However, spatio-temporal co-presence does not explicitly include the idea of interactions within individuals. Relations such as 'leading', 'following' or 'setting a trend' cannot be investigated by pure clustering alone.
▶ In essence, conventional spatial and spatio-temporal querying and clustering are inherently static and thus limited in their ability to cope with dynamic movement. Hence complementing techniques have to be explored in order to cope with the emerging new generation of movement data. Shirabe [54] illustrates such an alternative and uses correlation analysis in order to discover leader and follower relationships amongst moving individuals.

2.3 Promising Patterns

Precursory to this research Laube and colleagues proposed the REMO framework (RElative MOtion) which defines similar behaviour in groups of entities [36–38]. They defined a collection of movement patterns based on similar movement properties such as speed, acceleration or movement direction. Laube et al. [39] extended the framework by not only including movement properties, but also location itself. They defined several movement patterns, including flock (co-ordinately moving close together), trend-setter (anticipating a move of others), leadership (spatially leading a move of others), convergence (converging towards a spot) and encounter (meeting at a spot) and gave algorithms to compute them efficiently. Later Gudmundsson et al. [21] considered the same problems and extended the algorithmic results by primarily focusing on approximation algorithms – '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.' Benkert et al. [5] and Gudmundsson and van Kreveld [20] only recently revisited the flock pattern and gave a more generic definition that bases purely on the geometric arrangement of the moving entities and thus excludes the need of an analytical space as with the initial definition of the patterns [36, 39].

The model used in the REMO framework considers each time-step separately, that is, given $m \in \mathcal{N}$ and $r > 0$ a flock is defined by at least m entities within a circular region of radius r and moving in the same direction at some point in time. Benkert et al. [5] argued that this is not enough for many practical applications, e.g. a group of animals may need to stay together for days or even weeks before it is defined as a flock. They proposed the following definition of a flock:

Definition 1. *(m, k, r)-flock - Let $m, k \in \mathcal{N}$ and $r > 0$ be given constants. Given a set of n trajectories, where each trajectory consists of τ line segments, a flock in a time interval $I = [t_i, t_j]$, where $j - i + 1 \geq k$, 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.*

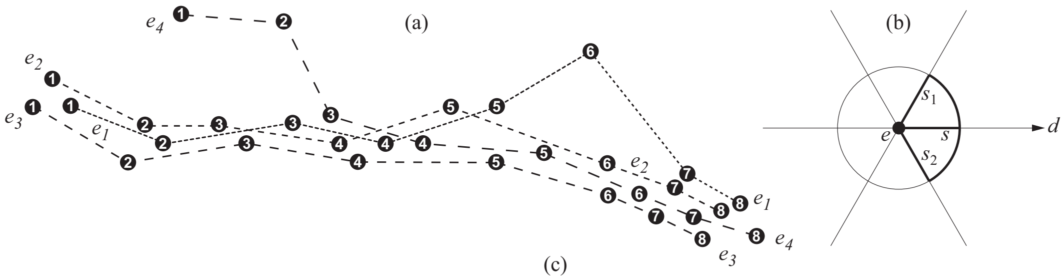
We will use a similar model when defining the leadership patterns, see Section 3. Using this model, Gudmundsson and van Kreveld [20] recently showed that computing the longest duration flock and the largest subset flock is NP-hard to approximate within a factor of $\tau^{1-\varepsilon}$ and $n^{1-\varepsilon}$, respectively, for any constant $\varepsilon > 0$. In the same model, Benkert et al. [5] described an efficient approximation algorithm for reporting and detecting flocks, where they let the size of the region deviate slightly from what is specified. Approximating the size of the circular region with a factor of $\Delta > 1$ means that a disk with radius between r and Δr that contains at least m objects may or may not be reported as a flock while a region with a radius of at most r that contains at least m entities will always be reported. Their main approach is a $(2 + \varepsilon)$-approximation (for any constant $\varepsilon > 0$) with running time $T(n) = O(kn(2^k \log n + k^2/\varepsilon^{2k-1}))$. Note that even though the dependency on the number of entities (namely n) is small, the dependency on the duration of the flock pattern (namely k) is exponential. Al-Naymat et al. [2] handle the problem of considering many entities and long-duration patterns by using a preprocessing step where the number of dimensions (i.e. time-steps) is reduced by random projection.
▶ A series of articles exploring simple flocking illustrated the potential of patterns based on the geometric arrangement of moving entities. The present paper shall achieve a similar definition for the more complex pattern leadership as well as efficient algorithms for its detection.

3 Leadership

We consider n entities moving in the two dimensional plane during the time interval $[t_1, t_\tau]$, see Figure 1(a) for an example. The infinite set T_p of time-points is defined as $T_p = \{t \mid t \in [t_1, t_\tau]\}$, and the set T_s of time-steps is the set of discrete time-points given as input, i.e. $T_s = \{t_1, ..., t_\tau\}$. We specify open and closed time intervals by (t_x, t_y) and $[t_x, t_y]$, respectively. A unit-time-interval is an open interval I between two consecutive time-steps, i.e. $I = (t_{x-1}, t_x)$, for a time-step t_x with $x > 1$.

3.1 Defining Leadership Patterns

For describing our leadership patterns, we need a couple of parameters specifying these patterns. More specifically, we assume that we are given numbers m (specifying the size of a pattern, i.e. the minimum number of entities involved in a pattern), k (the minimum temporal length of a pattern), a radius r (influencing the spatial size of a pattern), an angle α (also influencing the spatial size of a pattern) and an angle β (determining spatial characteristics of a pattern). We consider them as



(c)

IntervalsNotFwg(t):

Entity	1	2	3	4	5	6	7	8
e_1	0	1	2	0	1	2	3	0
e_2	0	0	0	0	0	1	0	0
e_3	0	0	0	0	0	0	0	0
e_4	0	1	2	3	4	5	6	7

IntervalsFwg(e',t):

Entity	e'	1	2	3	4	5	6	7	8
e_1	e_2	0	1	2	3	4	0	0	0
e_1	e_3	0	1	2	3	0	0	0	0
e_1	e_4	0	0	0	0	0	0	0	0
e_2	e_1	0	0	0	0	0	0	0	0
e_2	e_3	0	0	0	0	0	0	0	0
e_2	e_4	0	0	0	0	0	0	0	0
e_3	e_1	0	0	0	0	0	0	0	0
e_3	e_2	0	0	0	1	0	0	0	0
e_3	e_4	0	0	0	0	0	0	0	0
e_4	e_1	0	0	0	1	0	0	0	1
e_4	e_2	0	0	0	0	0	0	1	2
e_4	e_3	0	0	0	0	1	2	3	4

IntervalsFwd$_m$(t): *(m=1)*

Entity	1	2	3	4	5	6	7	8
e_1	0	1	2	3	4	0	0	0
e_2	0	0	0	0	0	0	0	0
e_3	0	0	0	1	0	0	0	0
e_4	0	0	0	1	2	3	4	5

numFws(t):

Entity	1	2	3	4	5	6	7	8
e_1	0	2	2	2	1	0	0	0
e_2	0	0	0	0	0	0	0	0
e_3	0	0	0	1	0	0	0	0
e_4	0	0	0	1	1	1	2	3

Fig. 1. (a) A set of 4 entities moving from left to right over 7 unit-time-intervals, i.e. over 8 time-steps. (b) Illustrating the definition of the front-region as the disc-segment within bold lines. (c) The follow-arrays of the four entities, where we use the front-region as depicted in (b) and $\alpha = \beta$.

constants during the rest of the paper, i.e. we will not carry them along as parameters of functions or other notations.

At time-point t_x, an entity e_j is located at a position with coordinates $xpos(e_j, t_x)$ and $ypos(e_j, t_x)$. As we do not have spatial information of an entity between two time-steps we make the following assumption for the remainder of this paper.

Assumption 1 *We assume that all entities move between two consecutive time-steps with constant direction and constant velocity.*

The same assumption has been used in earlier work [5]. It enables us to interpolate the positions of entities between time-steps. Even though we have no bound on the accuracy of this interpolation compared to the real positions of the entities, it appears to be a reasonable approach when tackling our leadership problems, as long as the sampling of points on the trajectories is sufficiently dense.

Suppose we are given an entity e_j at time-point t with $t_{x-1} < t < t_x$ for $t_x \in T_s$. We say e_j is heading into *direction* d where d is an angle in $[0, 2\pi)$ that is specified by the line segment e_j is moving along between time-steps t_{x-1} and t_x. (If e_j does not move between t_{x-1} and t_x then we define d to be the direction of the line segment e_j is moving along between the time-steps t_{x-2} and t_{x-1}. If no such time-steps exist, then we define $d := 0$.) The difference between two directions d_1 and d_2 is denoted by $||d_1 - d_2||$, and it is an absolute value, i.e. it is an angle in $[0, \pi]$. We declare the direction of an entity at a time-step t_x to be *undefined*, because at time-steps an entity might change its direction. However, the *direction* of an entity e_i at a time-step t_x *with respect to* (t_{x-1}, t_x) is the direction e_i is heading to at any time-point in (t_{x-1}, t_x). Therefore, when considering time intervals with certain properties of entities depending on direction, we implicitly exclude time-steps from those intervals in the remainder of this paper.

Given an entity e and a time-point $t \notin T_s$, we define the *front-region* of e at time t in the following way. Consider the disk C with radius r centred at $(xpos(e, t), ypos(e, t))$. Furthermore, consider three line segments s_1, s_2 and s of length r, all having one end point at $(xpos(e, t), ypos(e, t))$. Segment s points in the direction d that e is heading to at time t, and segments s_1 and s_2 are the well defined segments forming angles of $\frac{\alpha}{2}$ and $-\frac{\alpha}{2}$ with s, respectively. The part of the disk

C that contains s and is bounded by the segments s_1 and s_2 is the *front-region*, see Figures 1(b) and 2. We denote this wedge-shaped region by $front(e)$ *at time* t. An entity e_j is said to be *in front* of an entity e_i at time $t \notin T_s$ if and only if $e_j \in front(e_i)$ at time t.

Definition 2. *Let d_i and d_j be the directions of the entities e_i and e_j at time $t \notin T_s$, respectively. Entity e_i is said to* follow *entity e_j at time t, iff $e_j \in front(e_i)$ at time t and $||d_i - d_j|| \leq \beta$.*

An entity e_j is said to follow entity e_i at time $[t_x, t_y]$ for time-points t_x, t_y, if and only if e_i follows e_j at time t for all time-points $t \in [t_x, t_y] \setminus T_s$.

Definition 3. *An entity e_i is said to be a* leader *at time $[t_x, t_y]$ for time-points t_x, t_y, if and only if e_i does not follow anyone at time $[t_x, t_y]$, and e_i is followed by sufficiently many entities at time $[t_x, t_y]$. If there is an entity that is a leader of at least m entities for at least k time units, we have a* leadership pattern.

See Figure 2 for an example of some notations.

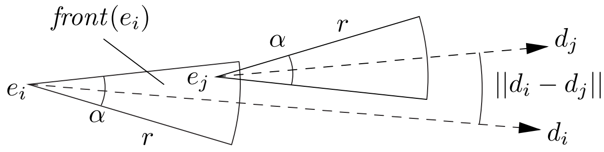


Fig. 2. The front regions of e_i and e_j as wedges of edge length r and apex angle α. Entity e_j is in front of e_i. The entities are heading into directions d_i and d_j, respectively. If $||d_i - d_j|| \leq \beta$ then e_i follows e_j.

Example 1. Consider the entities in Figure 1(a) where we use a front-region as depicted in Figure 1(b). We see that e_2 is following e_1 at time (t_1, t_5), e_1 is not following any other entity at time (t_1, t_3) and (t_4, t_7) and hence e_1 is a leader of e_2 at time (t_1, t_3) and (t_4, t_5).

In the remainder of this section, we consider two entities e_i and e_j and two consecutive time-steps t_{x-1} and t_x. The next lemma tells us that if we want to check whether an entity is following any other entity during the entire interval (t_{x-1}, t_x), we only have to check this at the two end points with respect to (t_{x-1}, t_x). The lemma is rather intuitive and can be proven with very much the same ideas as in the proof of Lemma 2 in [5].

Lemma 1. *Let e_i and e_j be two entities, and let t_{x-1} and t_x be two consecutive time-steps. If e_i follows e_j at time-points t_y and t_z with $t_{x-1} < t_y \leq t_z < t_x$ then under Assumption 1, e_i follows e_j at any time-point $t \in [t_y, t_z]$.*

Note that the lemma is also true for $t_{x-1} = t_y$ and $t_z = t_x$, however, the directions of the entities at these time-points are with respect to (t_{x-1}, t_x). Therefore, the time that an entity e_i follows another entity e_j between two consecutive time-steps t_{x-1} and t_x is a single subinterval of $[t_{x-1}, t_x]$, and such an interval can be computed in a straightforward manner.

Lemma 2. *Given two entities e_i and e_j and two time-steps t_{x-1} and t_x, we can compute in constant time the subinterval of $[t_{x-1}, t_x]$ for which $e_i \in front(e_j)$ and for which e_j follows e_i, under Assumption 1.*

3.2 Problem Statement

A leadership pattern exists if there is an entity that is a leader of sufficiently many entities over a long enough series of time-steps or time-points. Such a pattern is characterised by two values m which is the size of the set of followers, and k which is the length of a pattern. As mentioned in related work [5, 21] specifying exactly which of the patterns should be reported is often a subject for discussion. For instance, a leadership pattern of length exactly $k+1$ (starting at time-step t_x) implies the existence of two leadership patterns of length exactly k (albeit 'overlapping', one starting at time-step t_x and the other starting at time-step t_{x+1}). However, the pattern of length $k+1$ might be more interesting to report from a practical point of view. Therefore, we consider the following problems where we assume that m and k are constants.

- LP-report-all: For each entity e, report all time intervals where e is a leader of at least m entities for at least k time units.
- LP-max-length: Compute the length of a longest leadership pattern of size at least m, i.e. compute the largest value k^{max} such that there is an entity e that is a leader of at least m entities for k^{max} time units.
- LP-max-size: Compute the size of a largest leadership pattern of length at least k, i.e. compute the largest value m^{max} such that there is an entity e that is a leader of m^{max} entities for at least k time units.

All these problems come in four different flavours which are combinations of the modelling of the time axis (discrete vs. continuous) and the consistency of the set of followers (varying vs. non-varying).

More specifically, we consider each of the problems in a discrete case, where patterns (and follow behaviour) can only start and end at the discrete time-steps. In this discrete model, we can ensure that patterns exist, since we have the coordinates of the entities for all time-steps. Unlike this, patterns can start and end at any time-points in the continuous case. As discussed above, the data for the continuous case relies on Assumption 1. Recall that we do not have any guarantee on the accuracy of the linear interpolation between time-steps. This possible inaccuracy carries over to a possible inaccuracy of the reported leadership patterns in the continuous model. However, the continuous model is likely to become more important in the future, when huge data sets over many time-steps are available, which might need to be simplified in order to reduce storage space and processing time. Simplified trajectories are likely to be non-synchronous, yet they can approximate the original trajectory within a fixed specified error bound (see e.g. [6, 19]).

The other variation concerns the set of followers. If there is a subset S of entities such that for each time-point of the duration of the pattern all entities in S follow the leader (there may be additional followers as well at some time-points), then we call this a non-varying (subset) leadership pattern. In contrast to this, if we allow the subset of followers to change from one unit-time-interval to the next during the duration of the pattern (some entities may drop out, others may join in), then we call such a pattern a varying (subset) leadership pattern, as long as always at least m entities are following at each unit-time interval of the pattern. Depending on the application a non-varying or a varying set of followers might be desirable.

4 Algorithms for the Discrete Case

In the discrete case, patterns can only start and end at time-steps. We first describe arrays storing information about the follow behaviour of the entities with respect to a fixed entity e_i. Later, these arrays will be used to solve our leadership problems.

4.1 Getting Ready – Computing Follow-Arrays for an Entity e_i

For an entity e_i to determine whether it is a leader at the time (t_x, t_y), we need to know whether e_i is not following any other entity and whether e_i is followed by sufficiently many entities at (t_x, t_y). We consider e_i at this time as a potential leader, and we compute a couple of follow-arrays called '$\mathit{IntervalsNotFwg}(t_x)$', '$\mathit{IntervalsFwg}(t_x, e_j)$', '$\mathit{IntervalsFwd}_m(t_x)$' and '$\mathit{NumFws}(t_x)$'. The first three arrays store the number of consecutive unit-time-intervals that there is a certain follow-behaviour. In contrast to this, the fourth array stores the number of entities with a certain follow-behaviour.

IntervalsNotFwg: (short for 'the number of unit-time-intervals e_i is not following at t_x') The array *IntervalsNotFwg*(t_x) is a one dimensional array storing nonnegative integers. Such an integer for time-step t_x specifies for how many past consecutive unit-time-intervals (the last one ending at t_x) e_i is not following any other entity. That is, if *IntervalsNotFwg*$(t_x) = y$, then e_i is not following any other entity during the time interval (t_{x-y}, t_x). To compute the values of the *IntervalsNotFwg*-array, we use two nested loops. The outer loop runs from $t_x = t_2, ..., t_\tau$ (we start at $t_x = 2$ and set *IntervalsNotFwg*$(t_1) := 0$). The inner loop ranges over $e_j = e_1, ..., e_n$ and $e_j \neq e_i$. After each round of the inner loop we update *IntervalsNotFwg*(t_x) according to whether we found an entity e_j such that e_i follows e_j at time (t_{x-1}, t_x). According to Lemma 2 each such single test can be done in constant time.

IntervalsFwg: (short for 'the number of unit-time-intervals e_i is followed by e_j at t_x') The array *IntervalsFwg*(t_x, e_j) is a $(\tau \times n - 1)$ matrix storing nonnegative integers specifying for how many past consecutive unit-time-intervals (the last one ending at t_x) e_j is following e_i (with $e_j \neq e_i$). Filling the *IntervalsFwg*-array with the right values can also be done with two nested loops, one outer loop for $t_x = t_2, ..., t_\tau$ and one inner loop for $e_j = e_1, ..., e_n$ and $e_j \neq e_i$. Initially set *IntervalsFwg*$(t_1, e_j) := 0$. We test whether e_j follows e_i at the unit-time-interval (t_{x-1}, t_x), and if so, we update *IntervalsFwg*(t_x, e_j).

IntervalsFwd$_m$: (short for 'the number of unit-time-intervals e_i has at least m followers at t_x') The array *IntervalsFwd*$_m(t_x)$ is a one-dimensional array storing integers specifying for how many consecutive past unit-time-intervals (the last one ending at t_x) there are at least m entities following entity e_i. These m entities can be varying over time. Given the array *IntervalsFwg*, computing the *IntervalsFwd*$_m$-array can be done by looping over the *IntervalsFwg* array. We start at $t_x = 2$ and set *IntervalsFwd*$_m(t_1) := 0$. Now, we count in each column of *IntervalsFwg* (if we imagine the array *IntervalsFwg* to be arranged to have τ columns and $n - 1$ rows) the number of entities following e_i at the current time-step. If this number is smaller than m, we set *IntervalsFwd*$_m(t_x) := 0$, and if this number is at least m, we set *IntervalsFwd*$_m(t_x) :=$ *IntervalsFwd*$_m(t_{x-1}) + 1$.

NumFws: (short for 'the number of followers of e_i at t_x') Another array is *NumFws*(t_x) which is a one-dimensional array storing integers specifying how many entities are following entity e_i at time (t_{x-1}, t_x). Again, counting in each row of *IntervalsFwg* the number of entities following e_i at the current time-step yields the corresponding value of the *NumFws* array.

From the above discussion on the corresponding arrays, we conclude with the following lemma.

Lemma 3. *The IntervalsNotFwg, IntervalsFwg, IntervalsFwd$_m$ and NumFws-arrays for an entity e_i can be computed in $O(n\tau)$ time and space.*

Example 2. Consider the entities in Figure 1(a) where we use a front-region as depicted in Figure 1(b). Figure 1(c) shows four columns (one for each entity) of follow-arrays. To fill the arrays *IntervalsNotFwg* and *IntervalsFwg*, we need the trajectories and the front-regions. Once that is done, the arrays *IntervalsFwd*$_m$ and *NumFws* can be computed according to their definition.

4.2 Solving LP Problems with a Non-varying Subset of Followers

LP-report-all

In the discrete leadership version we assume that patterns can only start and end at time-steps $T_s = \{t_1, ..., t_\tau\}$. We use the arrays *IntervalsNotFwg* and *IntervalsFwg*, and we combine their information to determine whether e_i is a leader of a non-varying-subset of followers. To this end, we look for time-steps t_x such that *IntervalsNotFwg*$(t_x) \geq k$. For each such time-step t_x, we inspect the array *IntervalsFwg*(t_x, e_j) for $j = 1, ..., n$ and $j \neq i$, and we count the number of times that *IntervalsFwg*$(t_x, e_j) \geq k$. Let $m(k)$ denote this number. Now we can report e_i as a leader for every time-step t_x for which $m(k) \geq m$. As we only need to traverse our arrays at most once, this can be done in $O(n\tau)$ time.

Example 3. Let $k = 1$ and $m = 2$. Looking at the follow-arrays of entity e_1 in Figure 3, we see (shaded region) that e_1 is not following anyone, but is followed by 2 entities, and this happens for at least $k = 1$ unit-time-intervals at the time-steps 2 and 3. Hence, we would report two leadership-patterns with e_1 as leader.

So far, we have seen that we can compute in $O(n\tau)$ time and space at which time-steps an entity e_i is a leader. To find all leadership patterns amongst a set of entities we test any entity individually. As we only have to store one instance of each array at a time we can conclude with the following lemma.

Lemma 4. *Reporting all non-varying-subset leadership patterns of size at least m and length at least k, amongst n trajectories over τ time-steps can be done in $O(n^2\tau)$ time and $O(n\tau)$ space.*

		❶	❷	❸	❹	❺	❻	❼	❽
IntervalsNotFwg(t):	e_1	0	1	2	0	1	2	3	0
IntervalsFwg(e',t):	e_2	0	1	2	3	4	0	0	0
	e_3	0	1	2	3	0	0	0	0
	e_4	0	0	0	0	0	0	0	0

		❶	❷	❸	❹	❺	❻	❼	❽
IntervalsNotFwg(t):	e_2	0	0	0	0	0	1	0	0
IntervalsFwg(e',t):	e_1	0	0	0	0	0	0	0	0
	e_3	0	0	0	0	0	0	0	0
	e_4	0	0	0	0	0	0	0	0

		❶	❷	❸	❹	❺	❻	❼	❽
IntervalsNotFwg(t):	e_3	0	0	0	0	0	0	0	0
IntervalsFwg(e',t):	e_1	0	0	0	0	0	0	0	0
	e_2	0	0	0	1	0	0	0	0
	e_4	0	0	0	0	0	0	0	0

		❶	❷	❸	❹	❺	❻	❼	❽
IntervalsNotFwg(t):	e_4	0	1	2	3	4	5	6	7
IntervalsFwg(e',t):	e_1	0	0	0	1	0	0	0	1
	e_2	0	0	0	0	0	0	1	2
	e_3	0	0	0	0	1	2	3	4

Fig. 3. Follow-arrays with highlighted entries to mark patterns with a non-varying subset of followers.

LP-max-length

To compute the length of a longest pattern, where e_i is the leader, we utilise a variable k^{max}. Initially we set $k^{max} := 0$; we then loop once over all time-steps and at each time-step we may modify k^{max}, and at the end k^{max} will be equal to the length of a longest leadership pattern (for a specific m). Now, for each $t_x = t_1, ..., t_\tau$ we check whether $IntervalsNotFwg(t_x) > k^{max}$ and if so, we do the following. We inspect the column of the array *IntervalsFwg* corresponding to t_x. We traverse that column (i.e. we loop for $j = 1, ..., n,\ j \neq i$), and we count the number of entities e_j for which holds that $IntervalsFwg(t_x, e_j) > k^{max}$. Let this number be denoted by $m(k^{max})$. If $m(k^{max}) \geq m$, then we have at least m entities following e_i for more than k^{max} unit-time-intervals, and e_i is not following anyone during that time. Hence, we increase k^{max} by one and proceed with the next time-step t_{x+1}. Note that we only increase k^{max} by one as t_x is the first time-step for which $m(k^{max}) \geq m$. As we only traverse the entire arrays once, it takes $O(n\tau)$ time to compute the longest pattern, where e_i is the leader.

The following concluding lemma might surprise, as the longest duration flock pattern is NP-hard to compute and cannot even be approximated within a factor of $\tau^{1-\varepsilon}$ [20].

Lemma 5. *The longest duration leadership pattern for a non-varying-subset of followers of size at least m can be computed in $O(n^2\tau)$ time and $O(n\tau)$ space.*

Example 4. Consider again Figure 3. For $m = 1$, the above described method would find entity e_4 to be the leader (of one entity, namely e_3) for four consecutive unit-time-intervals, which is the length of a longest pattern (for $m = 1$).

LP-max-size

It is also possible to compute the size of a largest non-varying-subset of followers that follows a leader for at least k unit-time-intervals. We utilise the arrays *IntervalsNotFwg* and *IntervalsFwg* and a variable m^{max}, initially set to 0. We update this variable whenever we find a larger set of followers. That is, for $t_x := t_1, ..., t_\tau$, we test if both $IntervalsNotFwg(t_x) \geq k$ and $m(k) > m^{max}$, and if so, we set $m^{max} := m(k)$, where $m(k)$ is defined in the same way as $m(k^{max})$ in the section above. Hence, we obtain the following lemma.

Lemma 6. *The size of a largest non-varying-subset of entities that follow a leader for at least k time-steps can be computed in $O(n^2\tau)$ time and $O(n\tau)$ space.*

Example 5. Consider again Figure 3, and let $k = 1$. The algorithm above computes $m^{max} = 3$ as entity e_4 is a leader of 3 entities for $k = 1$ unit-time-interval at time-step 8.

4.3 Solving LP Problems with a Varying Subset of Followers

The variants of the problem of finding leadership patterns where the set of followers can change during the leadership pattern can be solved in a similar way as proposed in Section 4.2. To determine if an entity e_i is a leader of a varying-subset of followers, we use the follow-arrays $IntervalsNotFwg(t_x)$, $IntervalsFwd_m(t_x)$ and $NumFws(t_x)$ as described in Section 4.1.

LP-report-all

In the same flavour as described above, we can find out if e_i is a leader. We look for and report time-steps t_x, such that $IntervalsNotFwg(t_x) \geq k$ and $IntervalsFwd_m(t_x) \geq k$. It is easy to see that reporting when e_i is a leader can be done in $O(n\tau)$ time.

Example 6. Let $m = 1$ and $k = 2$. Consider e_1's follow-arrays in the upper half of Figure 4. Above method reports one time-steps (namely time-step 3) where e_1 is a leader of at least $m = 1$ entities for at least $k = 2$ unit-time-intervals.

The complexity of finding all leadership patterns for n entities is summarised as follows.

Lemma 7. *Reporting all varying-subset leadership patterns amongst n trajectories over τ time-steps can be done in $O(n^2\tau)$ time and $O(n\tau)$ space.*

		1	2	3	4	5	6	7	8
IntervalsNotFwg(t):	e_1	0	1	2	0	1	2	3	0
	e_2	0	0	0	0	0	1	0	0
	e_3	0	0	0	0	0	0	0	0
	e_4	0	1	2	3	4	5	6	7
IntervalsFwd$_m$(t): *(m=1)*	e_1	0	1	2	3	4	0	0	0
	e_2	0	0	0	0	0	0	0	0
	e_3	0	0	0	1	0	0	0	0
	e_4	0	0	0	1	2	3	4	5
IntervalsNotFwg(t):	e_1	0	1	2	0	1	2	3	0
	e_2	0	0	0	0	0	1	0	0
	e_3	0	0	0	0	0	0	0	0
	e_4	0	1	2	3	4	5	6	7
numFws(t):	e_1	0	2	2	2	1	0	0	0
	e_2	0	0	0	0	0	0	0	0
	e_3	0	0	0	1	0	0	0	0
	e_4	0	0	0	1	1	1	2	3

Fig. 4. Follow-arrays with highlighted entries to mark patterns with a varying subset of followers.

LP-max-length

For computing the longest duration leadership pattern, we use the arrays *IntervalsNotFwg* and $IntervalsFwd_m$, and we search for the largest k^{max} (initially $k^{max} := 0$) such that there is a time-step t_x for which $IntervalsNotFwg(t_x) \geq k^{max}$ and $IntervalsFwd_m(t_x) \geq k^{max}$. This can be done as follows. For $t_x = t_1, ..., t_\tau$, we check if $\min\{IntervalsNotFwg(t_x), IntervalsFwd_m(t_x)\} > k^{max}$, and if so, we perform an update $k^{max} := \min\{IntervalsNotFwg(t_x), IntervalsFwd_m(t_x)\}$ and proceed with the next time-step t_{x+1}.

Lemma 8. *The longest duration leadership pattern for a varying-subset of followers of size at least m can be computed in $O(n^2\tau)$ time and $O(n\tau)$ space.*

Example 7. Looking at the follow-arrays in the upper half of Figure 4, we see that e_4 is a leader of at least $m = 1$ entity for $k^{max} = 5$ unit-time-intervals (starting at time-step 3 and ending at time-step 8).

LP-max-size

If we would like to compute the size of a largest varying set of followers that follow e_i for at least k time-steps, we cannot use the array $\mathit{IntervalsFwd}_m$ directly as this array contains information only for one specific m. However, an easy way is to use binary search on m and recompute the $\mathit{IntervalsFwd}_m$ array for each value of m. This adds an additional $\log n$ factor to the running time.

We propose a slightly different approach. By spending linear preprocessing time, we can compute the minima of any substring of a sequence of numbers in $O(1)$ time. For more information on this Range Minimum Query (RMQ), see e.g. [4]. Now, we use the array *NumFws* and we look for at least k consecutive unit-time-intervals such that the minimum number of followers in the array *NumFws* during that time is as large as possible and e_i can be a leader. That is, we are looking for k consecutive unit-time-intervals such that e_i does not follow any other entity and for the largest minimum (to be referred to as m^{max}) over all numbers of followers corresponding to those k consecutive unit-time-intervals. All minima can be computed in $O(\tau)$ time [4], hence, m^{max} can be computed in linear time.

Lemma 9. *The size of a largest varying-subset of entities that follow a leader for at least k time-steps can be computed in $O(n^2\tau)$ time and $O(n\tau)$ space.*

Example 8. Consider the lower half of Figure 4 and let $k = 2$. The above algorithm computes $m^{max} = 2$ at time-step 3 for entity e_1 and at time-steps 8 for entity e_4.

5 Algorithms for the Continuous Case

In contrast to the discrete version of the leadership pattern, where a pattern can only start or end at the given discrete time-steps, in the continuous version of the problem a pattern can start and end at any point in time. As we do not have spatial information of the entities between two consecutive time-steps we use Assumption 1 to tackle the continuous version in this section. The main ideas are similar to the discrete case, but instead of using arrays storing single numbers to represent follow-behaviour we will use sets of time intervals. First, we describe how to compute them for a fixed entity e_i and then we define two operations on (sets of) intervals. Later, these intervals and operations are used to solve our leadership problems.

5.1 Getting Ready – Follow-Intervals for an Entity e_i

Computing Follow-Intervals: A first step is to compute a set *SetNotFwg* of notfollowing-intervals representing when a fixed entity e_i is not following any other entity e_j. An interval $I = (t_{x_a}, t_{y_a}) \in \mathit{SetNotFwg}$ with $t_{x_a} \leq t_{y_a}$ means that entity e_i is not following any other entity during the whole time interval I. Because entities move on a straight line between two consecutive time-steps, cf. Assumption 1, e_i can be involved in at most two events that change its follow-behaviour (i.e. the events of beginning or ending to follow) for each entity between two consecutive time-steps. That is why the set *SetNotFwg* contains $O(n\tau)$ intervals. We can compute this set with two nested loops one over all time-steps, another over all entities. By Lemma 2, this can be done in $O(n\tau)$ time in total.

We also need information about which entities follow e_i. This information is again stored in a set *SetFwd* of intervals. An interval $I = (t_{x_a}, t_{y_a}) \in \mathit{SetFwd}$ with $t_{x_a} < t_{y_a}$ means that e_i is followed by an entity, say e_j, during the whole time interval I. Also this set contains at most $O(n\tau)$ intervals, as an entity can change its follow behaviour with respect to e_i at most twice between two consecutive time-steps. We can compute this set with two nested loops one over all time-steps, another over all entities. By Lemma 2, this can be done in $O(n\tau)$ time in total.

Both sets of intervals can be computed in $O(n\tau)$ time. For the subsequent methods, however, we need the start- and end points of the intervals in non-decreasing order and that the intervals are maximal. Obtaining the sets such that the start- and end points are sorted can be done in $O(n\tau \log n)$ time in the following way. For each set we use two nested loops. The outer loop fixes an entity and the inner loop ranges over the time-steps. In that way it is easy to compute the intervals as maximal intervals. Whenever we compute start- or end points of an interval we can put them into $\tau - 1$ buckets, namely one for each unit-time-interval, i.e. pair of consecutive time-steps. As we can have at most $O(n)$ start- or end points in each bucket, we can sort all of them in all buckets in $O(n\tau \log n)$ time. Combining the sorted sequences of each bucket results in a sorted sequence of all start- and end points.

Lemma 10. *In $O(n\tau \log n)$ time and $O(n\tau)$ space, the sets SetNotFwg and SetFwd for an entity e_i can be computed such that all the start- and end points of the intervals in each set are output in non-decreasing order.*

Next, we define operations that take and return a set of intervals as input and output. We also briefly describe how to compute these operations, if the set of intervals is given along with the start- and end points in sorted order.

Combining Intervals: We call the first operation under consideration *interval-combination* denoted as $ic_x(S)$, where S is a set of intervals of $\mathbb{R}$. The operation outputs a set of non-intersecting intervals. Every point in $\mathbb{R}$ that is contained in at least x intervals of the input-set S will be in an interval of the output-set. Also, for every point that is contained in an interval of the output-set, there are at least x intervals in the input-set that all contain that point, see Figure 5. Note that $ic_1(S)$ is the union of all intervals in S and $ic_{|S|}(S)$ is the intersection of all intervals in S. Let S be a set of intervals where the start- and end points are given in sorted order. The operation $ic_x(S)$ can be computed by a parallel scan over the sorted start- and end points and keeping track of how many intervals are currently 'active'.

Lemma 11. *Suppose S is a set of intervals. If the start- and end points of the intervals in S are given in non-decreasing order, then we can compute $ic_x(S)$ in $O(|S|)$ time.*

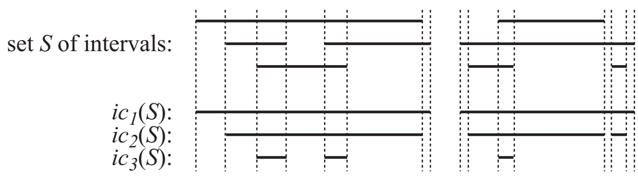


Fig. 5. The set S of intervals on the real line and the results after applying the ic_x operation for $x \in \{1, 2, 3\}$. Note that $ic_x(S) = \emptyset$ for all $x \geq 4$ as the intersection of any 4 intervals in S is empty.

Clipping Intervals: We also define another operation, which modifies single intervals. For an interval $I = \{t_{x_a}, t_{x_b}\}$, we cut or clip a part of length k at the beginning of I. If the resulting interval I' is non-empty, then that interval I' is the result of the operation. This operation can also be applied to all intervals of an entire set (cf. Figure 6), such that the order of the start- and end points of the intervals remains stable.

Lemma 12. *Let be given a set S of intervals, where the start- and end points of the intervals in S are given in non-decreasing order. We can compute $S' := \{I' \mid I' = clip_k(I), I \in S\}$ and output the start- and end points of all intervals in S' in non-decreasing order in $O(|S|)$ time.*

5.2 Solving LP Problems with a Non-Varying Subset of Followers

LP-report-all

We first look at the non-varying-subset version. In the previous section we have seen that we can compute the interval-set *SetFwd* in $O(n\tau \log n)$ time, where an interval in this set means

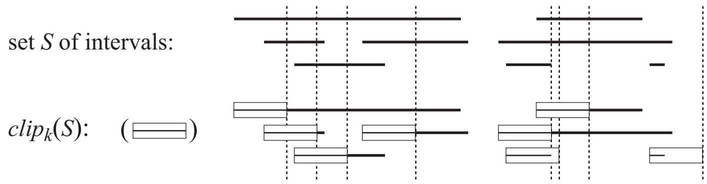


Fig. 6. The set S of intervals on the real line and the results after applying the $clip_k$ operation. For the $clip_k$ operation, the length of the interval in parentheses determines k.

that an entity follows e_i for the time of the interval. Now, we are going to modify the intervals in the set $SetFwd$. For each interval $I = \{t_{x_a}, t_{x_b}\} \in SetFwd$, we apply the operation $clip_k$ to obtain a set $SetFwd_{clipped} := \{I' \mid I' = clip_k(I), I \in SetFwd\}$. Note that $SetFwd_{clipped}$ (see Figure 7) only contains intervals whose originals had length at least k. The meaning of an interval $I' \in SetFwd_{clipped}$ is that there is an entity such that at each time-point $t \in I'$ this entity has already followed e_i for at least k time units (which is not necessarily the same as k unit-time intervals). The set $SetFwd_{clipped}$ can be computed in linear time with respect to the size of $SetFwd$, and this can be implemented such that the order of the (start- and end points of the) intervals remains stable.

We also clip the intervals of the set $SetNotFwg$ to obtain a set $SetNotFwg_{clipped} := \{I' \mid I' = clip_k(I), I \in SetNotFwg\}$ (see Figure 7). For each time-point in an interval in $SetNotFwg_{clipped}$, we have that e_i is not following any other entity for at least k time units.

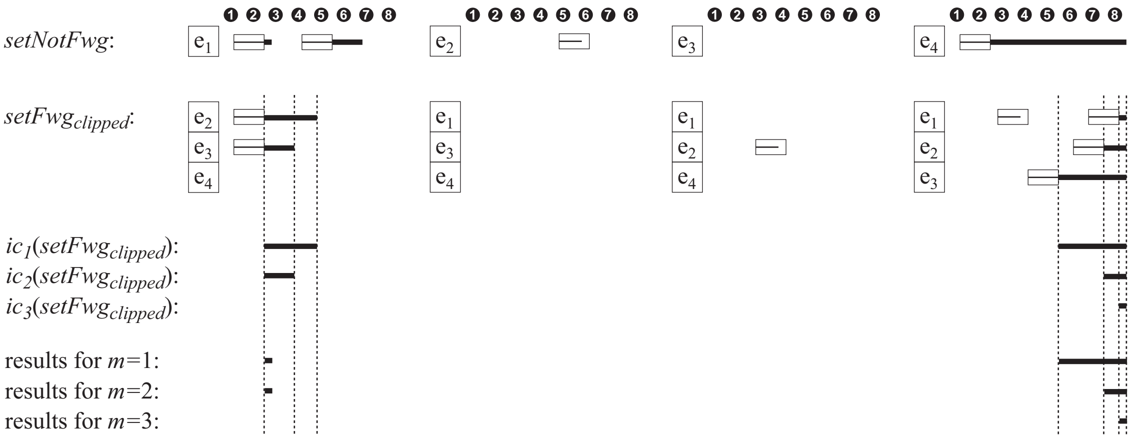


Fig. 7. Illustration of clipping and combining the intervals, where the intervals represent the follow-behaviour of the entities in Figure 1. The $result$-intervals are shown for different values of m.

The next step is to compute yet another set S of intervals as an interim result using one of the operations introduced in Section 5.1, $S := ic_m(SetFwd_{clipped})$. For any time-point in an interval in S there are at least m entities following e_i, where each of those entities already followed e_i for at least k time units. Finally, we combine the information represented by S and $SetNotFwg_{clipped}$. What we need is similar to a logical 'and' between intervals of those two sets, and this can be done by applying the ic_x again to obtain a set of $result$-intervals, $result := ic_2(S \cup SetNotFwg_{clipped})$. Note that the start- and end points of the set $S \cup SetNotFwg_{clipped}$ can be sorted in linear time if the start- and end points of S and $SetNotFwg_{clipped}$ are sorted. The set $result$ contains all intervals for which e_i is a leader of at least m entities for at least k time units. If we would like to report the leadership patterns of all entities, we apply the above method to each entity. Hence, we can conclude with the following lemma.

Lemma 13. *Let be given n trajectories over τ time-steps. Reporting all time intervals where there is an entity a leader of a non-varying subset of at least m entities for at least k time units, can be done in $O(n^2\tau \log n)$ time and $O(n\tau)$ space.*

LP-max-size

We can use the sets $SetNotFwg_{clipped}$ and $SetFwd_{clipped}$, where the intervals are given in non-decreasing order, to find the maximum m^{max} for which e_i is a leader of a non-varying set of m^{max} entities for at least k time units. To that end, we do not collapse the set $SetFwd_{clipped}$ into a set S as described above, but we utilise a parallel scan over the intervals in $SetNotFwg_{clipped}$ and $SetFwd_{clipped}$.

By a parallel scan we mean moving an imaginary vertical line over the horizontally arranged intervals, stopping at certain points and performing certain actions. In our case the points where we stop are the start- and end points of the intervals. For any position of the scan-line we say an interval I is *active*, if the scan-line ℓ intersects interval I.

During the parallel scan, we keep track of the number of active intervals in $SetFwd_{clipped}$, where the intervals in $SetNotFwg_{clipped}$ are used as a mask (see Figure 8). All this can be done in $O(n\tau)$ time.

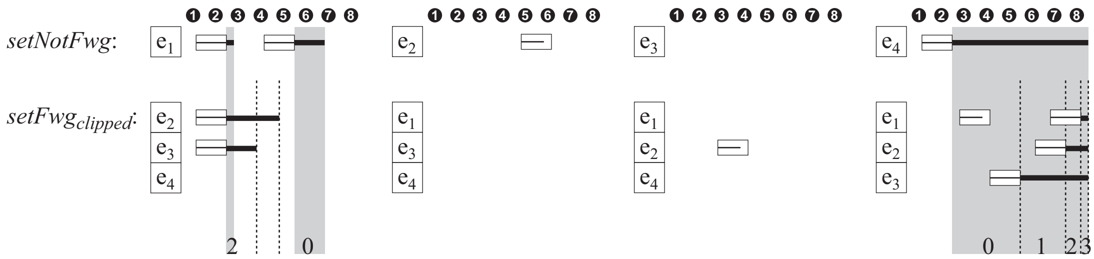


Fig. 8. Illustrating the parallel scan approach. The shaded region indicates how the *SetNotFwg* intervals are used as a mask. The numbers indicate the number of active *SetFwd* intervals.

Lemma 14. *Let be given n trajectories over τ time-steps. Computing the maximum size of a non-varying subset of followers which follow a leader for at least k time units, can be done in $O(n^2\tau \log n)$ time and $O(n\tau)$ space.*

LP-max-length

A method similar to the one presented above cannot be used directly, as the sets of intervals are computed for specified values of k. We could use binary search on k, however, this would add another $\log \tau$ factor to the running time. The method described in this section also builds upon the sets *SetNotFwg* and *SetFwd* of intervals, introduced in Section 5.1. It finds the largest k^{max}, such that there is a non-varying subset of at least m entities following entity e_i for k^{max} time-units. However, it can also be used to report patterns where e_i is a leader of at least m non-varying followers for at least k time-steps. We do this by performing a parallel scan over the sets of intervals, assuming they are given such that the start- and end points of the intervals are in non-decreasing order.

During the parallel scan we keep track of the active intervals in *SetNotFwg*. Note that only one interval $I \in SetNotFwg$ can be active at a time. By keeping a pointer p_1 to I, we know for every time-point t, whether e_i is following any other entity. If e_i is following any other entity, then there is no interval in *SetNotFwg* active at time t (and p_1 becomes a null-pointer). On the other hand, if there is an interval $I \in \mathit{SetNotFwg}$ active at t, then we can compute for how long e_i is not following any other entity.

We also keep track of how many entities follow e_i and for how long. To this end, let t be a time-point during the parallel scan. Let $A \subseteq \mathit{SetFwd}$ be the set of all intervals in *SetFwd* that are active at time t. We will not maintain A, but only a variable m' with $m' = |A|$ (initially $m' := 0$). Furthermore, we will maintain a pointer p_2 to the interval in A with the m-th largest end point. If A contains less than m intervals, then p_2 points to some interval. (As we will not use pointer p_2 if A contains less than m intervals it is not important where p_2 points to in that case.)

Before the parallel scan, we initialise $k^{max} := 0$, and after the parallel scan k^{max} will be the length of the longest leadership pattern with e_i as leader of a non-varying set of at least m entities. We also introduce an artificial interval which starts and ends before any other interval starts and we initialise p_2 to point to that interval. This interval is introduced merely to have pointer p_2 well initialised. The parallel scan does not take this interval into consideration. As mentioned above the points where we stop with the scan-line are the start- and end points of the intervals, and if two such points have the same time, we process them one after the other, as if one was infinitesimally later than the other. By maintaining all invariants it is easy to see that for every position of the scan-line with corresponding time t, we can check if there are at least m entities following e_i, i.e. if $m' \geq m$. In the case that there are at least m followers of e_i, we also can determine for how long in the future all these entities will follow e_i, by using the pointer p_2. Furthermore, we can check if e_i is following any other entity (by using p_1), and if not for how long in the future e_i will not follow any other entity. Therefore, we can determine whether there is a leadership pattern, with e_i as leader of a non-varying set of at least m entities, and if there is such a pattern, we can also determine its length k'. If $k' > k^{max}$ then we perform an update $k^{max} := k'$.

By doing this parallel scan approach for each entity, we can compute the overall longest duration leadership pattern.

Lemma 15. *Let be given n trajectories over τ time-steps. Computing the maximum length of a leadership pattern with a non-varying subset of followers of size at least m, can be done in $O(n^2\tau \log n)$ time and $O(n\tau)$ space.*

5.3 Solving LP Problems with a Varying Subset of Followers

LP-report-all

After considering the case for the non-varying subset in Section 5.2, the case for a varying subset is rather easy. Here, we do not require that all entities follow e_i for k time-units. Hence, with the terminology as used before we compute a set $S := ic_m(\mathit{SetFwd})$. For any time-point in an interval in S there are at least m entities following e_i. As e_i still has to be followed for at least k time-units, we clip all intervals in S to obtain $S' := \{I' \mid I' = \mathit{clip}_k(I), I \in S\}$. As before, our last step is to combine S' and $\mathit{SetNotFwg}_{clipped}$ to obtain the set of *result*-intervals, $\mathit{result} := ic_2(S' \cup \mathit{SetNotFwg}_{clipped})$.

Lemma 16. *Let be given n trajectories over τ time-steps. Reporting all time intervals where there is an entity a leader of a varying subset of at least m entities for at least k time units, can be done in $O(n^2\tau \log n)$ time and $O(n\tau)$ space.*

LP-max-size

In this case, we can use the approach mentioned in Section 4.3, where we spend additional time for binary search on m to find m^{max}.

Lemma 17. *Let be given n trajectories over τ time-steps. Computing the maximum size of a varying subset of followers which follow a leader for at least k time units, can be done $O(n^2\tau \log n)$ time and $O(n\tau)$ space.*

LP-max-length

To find the length of a longest duration leadership pattern of a varying set of at least m entities, we can use a similar approach as in Section 5.3. We also compute a set $S := ic_m(SetFwd)$, such that for each time-point in an interval $I \in S$, we know that there are at least m followers of e_i. To combine this with the information when e_i is not following any other entity, we apply the operation ic_x once again to obtain $result := ic_2(S \cup SetNotFwg)$. Now, for any interval in $result$ it holds that e_i is not following any other entity, and also that e_i is followed by at least m entities. Searching for the length of the longest interval in $result$ solves the problem at hand for entity e_i.

Lemma 18. *Let be given n trajectories over τ time-steps. Computing the maximum length of a leadership pattern with a varying subset of followers of size at least m, can be done $O(n^2\tau \log n)$ time and $O(n\tau)$ space.*

5.4 Hardness in the Continuous Case

It is likely that every algorithm for the continuous version of the leadership problem that detects leadership patterns between two consecutive time-steps in a set of n trajectories requires $\Omega(n^2)$ time in the worst case. This can be shown by a transformation from the problem Point-on-3-lines, which was proven to be 3-sum-hard [18]. There is no subquadratic time algorithm known for those problems. For a weak model of computation a lower bound of $\Omega(n^2)$ for those problems exists [15]. We can conclude with the following lemma (see [3] for more details).

Lemma 19. *Finding continuous leadership patterns between two consecutive time-steps in a set of trajectories is* 3-sum-*hard.*

6 Experimental Evaluation

This section is devoted to reporting the experimental results. The algorithms were implemented in Java[4] and all experiments were performed on a Linux operated PC with an Intel 3.6 GHz processor and 2 GB of main memory.

6.1 Input Data

All input files were generated artificially with NetLogo [62]. More specifically, we modified NetLogo's Flocking Model [61] such that entities do not wrap around the world-borders, but will be repulsed smoothly from walls, see Figure 9. Furthermore, we added some code for moderate random changes in an entity's direction and saving the coordinates into a file. There are many parameters to modify the behaviour of the entities and thus also to modify how many flocks and leadership-patterns are created. However, we have no direct control over the exact number or length or size of patterns.

We generated files with variable number of entities (128-4096), two different sizes of the underlying universe U (i.e. coordinate space 512×512 and 1024×1024) and two different characteristics CH (i.e. $CH = u$ and $CH = c$) of the entity distribution. $CH = u$ means that the parameters of the Flocking Model were chosen such that the entities are more *u*niformly distributed, i.e. only small clusters emerge. Flocks (and thus leadership patterns) still exist but their size and length are likely to be smaller than those of the other characteristic. $CH = c$ means that the parameters of the Flocking Model were chosen such that the entities form few but rather large *c*lusters, and hence, the flocks tend to contain more entities and have a longer duration. The number of time-steps is $\tau = 1000$.

[4] Java was chosen because this increases the platform independence and it makes it easier to integrate the code into an existing larger framework.

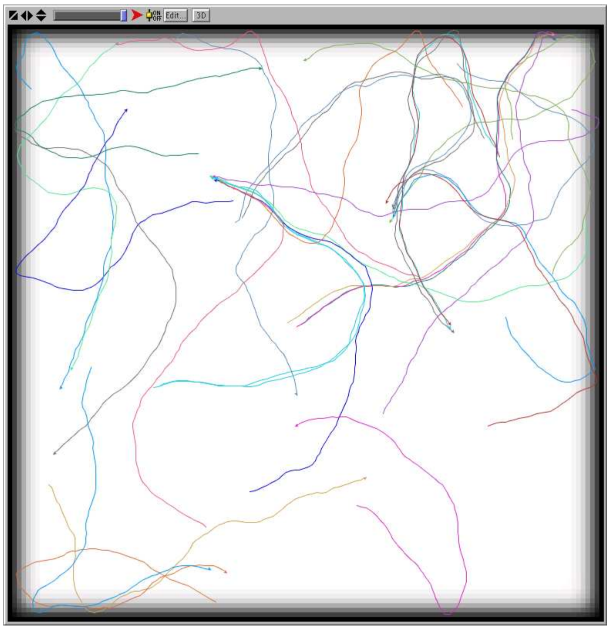


Fig. 9. This screenshot of NetLogo's modified Flocking model shows the trajectories of 32 entities in a universe with side lengths 128×128, run for 100 time-steps.

6.2 Methods

We performed experiments with two variants of our algorithms for the discrete case. The first one is a straight forward implementation of the method described in Section 4. This method contains (among others) two nested loops ranging over all entities. The disadvantage from a practical point of view is that when looking for entities that might be in a front-region, then also entities that are too far away will be considered. Therefore, our second method tries to overcome this drawback by dividing the underlying plane into buckets (squares of side length r). Now when looking for entities that might be in a front-region, only those entities will be considered that are in the nine neighbouring buckets (including the bucket at the centre). Note that for each of our leadership problems, all methods always compute all arrays from scratch. Especially the arrays *IntervalsNotFwg* and *IntervalsFwg* could be used three times after computing them once. For an easier comparison however, we refrained from doing so.

6.3 Results

Tables 1 and 2 show the results of our algorithms for $m = 10$, $k = 20$, $r = 20$, $\alpha = \pi$ and $\beta = \frac{\pi}{2}$. From our point of view the running times and their asymptotic behaviour are much more interesting than for example the exact number of patterns found as we deal with artificial data. Nevertheless, in Table 1 we can see how many entities have been leaders (leaders), the number of leadership patterns found (report-all), the length of a longest duration pattern (max-length) and the number of entities in a pattern with most followers (max-size). Note that patterns with length $> k$ will be reported multiple times as patterns of length k.

We observed that the vast majority of the running time is spent on computing the arrays *IntervalsNotFwg* and *IntervalsFwg* (which can be done in $O(n^2\tau)$ time). Once these two arrays are computed, computing more arrays and/or extracting information to solve the leadership problem is very efficient (linear time). Therefore, our methods for the three different leadership problems result almost always in the same running times (they differ on average less than three percent), as they compute all arrays from scratch. Hence, Table 2 depicts the running times of our methods only for the report-all leadership problem.

n	U	CH	non-varying				varying			
			leaders	report-all	max-length	max-size	leaders	report-all	max-length	max-size
128	512^2	c	2	89	37	23	2	161	41	23
256	512^2	c	10	211	56	66	12	389	71	71
512	512^2	c	11	329	44	200	13	520	68	238
1024	512^2	c	27	676	46	197	31	1142	66	208
2048	512^2	c	33	689	57	384	36	1022	72	419
4096	512^2	c	44	966	55	812	50	1541	78	959
128	1024^2	c	0	0	0	4	0	0	0	4
256	1024^2	c	0	0	2	8	0	0	2	8
512	1024^2	c	19	360	46	26	27	643	60	26
1024	1024^2	c	36	954	53	166	47	1591	73	183
2048	1024^2	c	80	1536	47	219	97	2833	101	224
4096	1024^2	c	98	2521	59	257	117	4299	78	324
128	512^2	u	0	0	0	3	0	0	0	4
256	512^2	u	0	0	9	7	0	0	13	7
512	512^2	u	1	7	25	10	6	54	41	13
1024	512^2	u	5	15	24	13	9	73	32	21
2048	512^2	u	8	40	34	15	29	187	40	25
4096	512^2	u	6	36	29	14	19	109	34	37
128	1024^2	u	0	0	0	3	0	0	0	3
256	1024^2	u	0	0	0	5	0	0	0	5
512	1024^2	u	0	0	0	7	0	0	0	7
1024	1024^2	u	1	1	20	10	1	2	20	10
2048	1024^2	u	6	26	25	11	20	152	40	17
4096	1024^2	u	16	87	42	24	49	279	42	24

Table 1. Resulting values of our methods.

n	U	CH	without buckets		with buckets	
			non-varying	varying	non-varying	varying
128	512^2	c	9.44	9.56	2.35	2.86
256	512^2	c	40.91	41.89	12.58	14.69
512	512^2	c	203.53	212.88	102.74	119.15
1024	512^2	c	664.01	683.83	191.85	221.47
2048	512^2	c	3393.17	3457.26	972.34	1099.19
4096	512^2	c	14903.81	15046.69	5250.53	5651.59
128	1024^2	c	8.19	8.47	0.91	1.24
256	1024^2	c	32.79	33.82	2.83	3.63
512	1024^2	c	132.97	139.13	12.00	15.01
1024	1024^2	c	595.24	622.29	110.06	129.27
2048	1024^2	c	2809.12	2875.07	324.43	375.57
4096	1024^2	c	11143.28	11300.18	1477.70	1705.00
128	512^2	u	8.37	8.08	1.33	1.52
256	512^2	u	32.73	32.96	3.74	4.67
512	512^2	u	129.16	130.56	12.88	15.91
1024	512^2	u	529.79	523.12	47.54	54.52
2048	512^2	u	2184.42	2178.64	221.99	234.74
4096	512^2	u	11126.42	10978.71	1024.22	1037.39
128	1024^2	u	7.85	7.86	0.87	1.04
256	1024^2	u	31.45	32.79	2.28	3.01
512	1024^2	u	127.92	128.21	7.47	8.80
1024	1024^2	u	512.59	515.91	24.67	27.96
2048	1024^2	u	2268.77	2251.06	89.00	95.24
4096	1024^2	u	11201.63	11295.04	350.20	381.82

Table 2. Running times of our methods for the report-all problem. Reported times are in seconds.

6.4 Observations

Non-Varying vs. Varying: As we could expect running times for the patterns with a varying subset of followers are often higher, as one more array is computed for the 'varying' problems. However, this increase is very marginal compared to other influencing factors. We can also observe that the values for the 'varying' patterns are at least as big (sometimes slightly larger) as for the 'non-varying' patterns. This is because a non-varying pattern is also a varying pattern by definition.

Without Buckets vs. With Buckets: The approach to subdivide the space into buckets does not influence the reported values of our methods, however, it can have an impressive impact on the running times (see Figures 10 and 11). Depending on the input characteristics, we can observe speed-up factors between 2 and 32. The running time of the methods without 'buckets' is clearly quadratic in the number of entities. An asymptotic behaviour of the methods with 'buckets' is more difficult to identify, but note that also this method has a quadratic worst case running time.

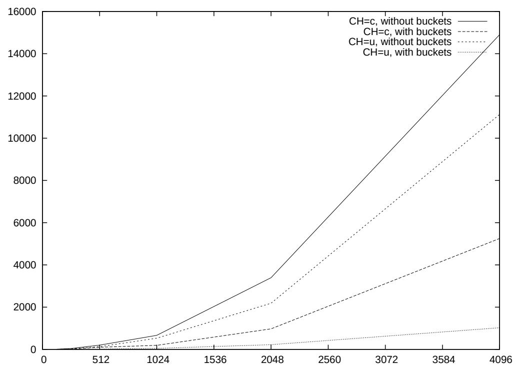


Fig. 10. Running times depending on input size for non-varying report-all patterns for $U = 512^2$.

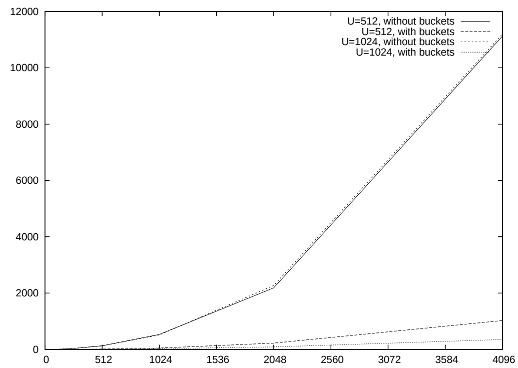


Fig. 11. Running times depending on input size for non-varying report-all patterns for $CH = u$.

$CH = u$ vs. $CH = c$: Almost always the input files with characteristic $CH = c$ contain more patterns, longer patterns and larger patterns, which was expected as those files are much more likely to contain more and larger flocks. Hence, the input files with $CH = u$ result in smaller running times (see also Figure 10). Interestingly these characteristics also indicate that the 'bucket' approach for speeding-up the computations has its limitations, because the speed-up factor of the 'buckets' method is strongly influenced by the characteristics. For $CH = u$, we observe speed-up factors around between 5 and 11 for the instances with $U = 512^2$, and between 7 and 32 for the instances with $U = 1024^2$. On the other hand, for $CH = c$, the speed-up factors are between 2 and 4 for the instances with $U = 512^2$, and between 5 and 11 for the instances with $U = 1024^2$. This can be explained by noting that the files with characteristic $CH = c$ contain more and bigger flocks, and hence it is more likely that our algorithms encounter neighbouring buckets that are filled with more entities.

$U = 512^2$ vs. $U = 1024^2$: The difference between the universe with $U = 512^2$ and $U = 1024^2$ is that the former is much denser when filled with the same number of entities. As a result, in the larger universe ($U = 1024^2$) less and smaller patterns exist. Also the running times are affected (see Figure 11). The methods with buckets run faster on instances with a larger universe, because we have more buckets and therefore, buckets are likely to contain less entities on average.

7 Discussion

The analysis of the interrelations of moving individuals has in the last five years attracted increasing attention, as a general reaction to the striking need for more powerful methods for surveillance and geospatial intelligence. Geographical information scientists are commissioned to develop methods that detect the expected and discover the unexpected from massive streams of disparate data, potentially originating from various sources [58]. Such methods need to be *scalable*, *flexible* and *reliable*. This section discusses our leadership approach with respect to these three properties and discusses the found algorithm running times.

Balancing the matching of formalised movement patterns (such as the presented leadership) with the inferring of unexpected space-time behaviours from visualised space-time paths, we argue that the former copes much better with increasing data sets. Whereas inferring form visualisation might be adequate for the analysis of individual events of interest [30], keeping track of hundreds of individuals cruising in the space-time aquarium is literally impossible [34]. By contrast, when detecting movement patterns such as flock or leadership, the number of entities n is just a performance factor but not an obfuscation factor.

Approaches detecting leader and follower relationships using pair-wise cross-correlation of trajectories suffer from their intrinsic limitation to very small numbers of involved entities. Thus, lead-follow events in [54], for example, can only be detected for pairs of individuals at a time. Our leadership pattern, in contrast, allows individuals to lead groups of followers. Since they operate on local-instantaneous events they can be detected in trajectories of variable lengths, as long as there is certain temporal overlap. Furthermore the approach in [54] has rather demanding constraints with respect to the analysed data set. It requires trajectories of equal length and strongly synchronous sampling. Even though we assume the input data to have the same characteristics, our algorithms for the continuous case can be easily applied to data without a synchronised sampling. The running times for sorting the sets of intervals for an entity would slightly increase, however, from $O(n\tau \log n)$ to $O(n\tau \log n\tau)$. We argue that our leadership algorithms are thus flexible and applicable to diverse data from various sources.

Movement patterns that are defined from the geometric arrangement of the involved entities (e.g. leadership), are more reliable than movement patterns that base on the intermediate step of an analysis matrix, as do the REMO patterns depend on an analysis matrix in Laube et al. [37]. The deterministic discretisation of the movement descriptors in eight cardinal direction classes introduces edge effects. An example shall illustrate such edge effects. Let $22.5°$ be one threshold of the discrete movement azimuth class 'North'. Let furthermore the pattern under study be a flock pattern of four entities moving in the same direction at any time t. Why should a set of entities S_1 with azimuths $[21°, 22°, 22°, 21°]$ be a flock when another set S_2 with azimuths $[22°, 23°, 23°, 22°]$ is not? A definition requiring the entities to have a mean azimuth and some variance (e.g. $\pm 22.5°$) is a much more natural and thus reliable definition of flock. The definition of leadership in this paper follows for exactly the same reasons the road of using a geometrical arrangement instead of scanning a discretised matrix.

When comparing the running times in this article with those reported in [5], we observe that the running times in the present work are much higher. This is because the used methods are different. The methods in [5] are faster but only report patterns of a specified length with a specified start- and end-time. The methods in this paper, however, are more flexible. Once the arrays *IntervalsNotFwg* and *IntervalsFwg* are computed we can very efficiently use them to report patterns of different lengths, and with different start- and end-times. We also developed and implemented an approximation algorithm and performed initial experiments. They show a better asymptotic behaviour of the approximation algorithm. However, the constant factors seem to be too large for practical purposes, because for our test-files the exact algorithms always outperformed the approximation algorithm. More details on this algorithm can be found in [3].

8 Conclusions and Outlook

Movement patterns detect structure in large tracking data sets and are thus key to a better understanding of the interactions amongst moving agents. We provide a formal description of the pattern 'leadership' and subsequently algorithms for its efficient detection. 'Leadership' describes the event or process of one individual in front leading the movement of a group. Our approach is inspired by movement patterns documented in the animal behaviour and behavioural ecology literature.

Our experiments give indications which input-size can be processed within a reasonable amount of time, and they have shown that we are able to efficiently report leadership patterns. The resulting running times match the theoretical bounds, however for improved methods (with buckets) the running times strongly depend on the characteristics of the instance.

In this article we assumed that all the trajectories fit into main memory. If this is not the case then we would have to develop I/O-efficient algorithms or use spatio-temporal index structures. Both these techniques would probably improve performance if the input does not fit into main memory. However, this is an extension that would require much more future research.

One drawback of the given definition of leadership is that a leader has to be in the front region of all followers. For instance, for a very big flock of gnus this definition might not be applicable, as some gnus at the end of the flock are too far away from the front-line to be able to see leading animals. Hence, one direction for future research could be the definition and analysis of cascading leaders or followers, where a cascading follower is a follower of a leader or a follower of another cascading follower.

For the many fields interested in movement, the overall challenge lies in relating movement patterns with the surrounding environment, in order to understand *where*, *when* and ultimately *why* the agents move the way they do. Conceptualising detectable movement patterns and the development of algorithms for their detection is a first important step towards this ambitious long-term goal. With its traditional spatial awareness, computational geometry can make immense contributions to the theoretical framework underlying movement analysis in geographical information science, behavioural ecology or surveillance and security analysis.

Acknowledgements

The authors wish to thank Karin Schütz, AgResearch Ruakura, Hamilton, New Zealand for valuable commets on animal movement patterns, Bojan Djordjevic for implementing the algorithms and the anonymous reviewers of this and earlier versions of this article.

References

1. P. K. Agarwal, L. Arge, and J. Erickson. Indexing moving points. *Journal of Computer and System Sciences*, 66(1):207–243, 2003.
2. G. Al-Naymat, S. Chawla, and J. Gudmundsson. Dimensionality reduction for long duration and complex spatio-temporal queries. In *Proceedings of the 22nd ACM Symposium on Applied Computing*, pages 393–397. ACM, 2007.
3. M. Andersson, J. Gudmundsson, P. Laube, and T. Wolle. Reporting leaders and followers among trajectories of moving point objects. Technical Report PA006075, National ICT Australia, 2006. `http://www.nicta.com.au`, Extended abstract in *Proceedings of the 22nd ACM Symposium on Applied Computing*, pages 3–7. ACM, 2007.
4. M. A. Bender and M. Farach-Colton. The LCA problem revisited. In *LATIN '00: Proceedings of the 4th Latin American Symposium on Theoretical Informatics*, volume 1776 of *Lecture Notes In Computer Science*, pages 88–94, London, UK, 2000. Springer-Verlag.
5. M. Benkert, J. Gudmundsson, F. Hübner, and T. Wolle. Reporting flock patterns. In *Proceedings of the 14th European Symposium on Algorithms (ESA 2006)*, volume 4168 of *Lecture Notes in Computer Science*, pages 660–671. Springer, 2006.
6. H. Cao, O. Wolfson, and G. Trajcevski. Spatio-temporal data reduction with deterministic error bounds. *The VLDB Journal*, 15(3):211–228, 2006.

7. S. M. C. Cavalcanti and F. F. Knowlton. Evaluation of physical and behavioral traits of llamas associated with aggressiveness toward sheep-threatening canids. *Applied Animal Behaviour Science*, 61(2):143–158, 1998.
8. N. R. Chrisman. Beyond the Snapshot: Changing the Approach to Change, Error, and Process. In M. J. Egenhofer and R. G. Golledge, editors, *Spatial and Temporal Reasoning in Geographic Information Systems*, pages 85–93. Oxford University Press, Oxford, UK, 1998.
9. L. Conradt and T. J. Roper. Group decision-making in animals. *Nature*, 421(6919):155–158, 2003.
10. M. D'Auria, M. Nanni, and D. Pedreschi. Time-focused density-based clustering of trajectories of moving objects. In *Proceedings of the Workshop on Mining Spatio-temporal Data (MSTD-2005)*, Porto, 2005.
11. C. Du Mouza and P. Rigaux. Mobility patterns. *GeoInformatica*, 9(4):297–319, 2005.
12. B. Dumont, A. Boissy, C. Achard, A. M. Sibbald, and H. W. Erhard. Consistency of animal order in spontaneous group movements allows the measurement of leadership in a group of grazing heifers. *Applied Animal Behaviour Science*, 95(1-2):55–66, 2005.
13. J. A. Dykes and D. M. Mountain. Seeking structure in records of spatio-temporal behaviour: visualization issues, efforts and application. *Computational Statistics and Data Analysis*, 43(4):581–603, 2003.
14. N. Eagle and A. Pentland. Reality mining: sensing complex social systems. *Personal and Ubiquitous Computing*, 10(4):255–268, 2006.
15. J. Erickson and R. Seidel. Better lower bounds on detecting affine and spherical degeneracies. *Discrete & Computational Geometry*, 13:41–57, 1995.
16. M. Erwig and R. H. Güting. Spatio-Temporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases. *GeoInformatica*, 3(3):269–296, 1999.
17. A. U. Frank. Socio-Economic Units: Their Life and Motion. In A. U. Frank, J. Raper, and J. P. Cheylan, editors, *Life and motion of socio-economic units*, volume 8 of *GISDATA*, pages 21–34. Taylor & Francis, London, 2001.
18. A. Gajentaan and M. H. Overmars. n^2-hard problems in computational geometry. Technical Report 1993-15, Department of Coumputer Science, Utrecht University, The Netherlands, 1993.
19. J. Gudmundsson, J. Katajainen, D. Merrick, C. Ong, and T. Wolle. Compressing spatio-temporal trajectories. Manuscript, July 2007.
20. J. Gudmundsson and M. van Kreveld. Computing longest duration flocks in trajectory data. In *Proceedings of the 14th ACM Symposium on Advances in GIS*, pages 35–42, 2006.
21. J. Gudmundsson, M. van Kreveld, and B. Speckmann. Efficient detection of motion patterns in spatio-temporal sets. *GeoInformatica*, 11(2):195–215, 2007.
22. S. Gueron and S. A. Levin. Self-Organization of Front Patterns in Large Wildebeest Herds. *Journal of theoretical Biology*, 165(4):541–552, 1993.
23. R. Güting, M. H. Boehlen, M. Erwig, C. S. Jensen, N. Lorentzos, E. Nardelli, M. Schneider, and M. Vazirgiannis. A Foundation for Representing and Querying Moving Objects. *ACM Transactions on Database Systems (TODS)*, 2520(1):1–42, 2000.
24. R. Güting, M. H. Boehlen, M. Erwig, C. S. Jensen, N. Lorentzos, E. Nardelli, M. Schneider, and J. R. R. Viqueira. Spatio-temporal Models and Languages: An Approach Based on Data Types. In M. Koubarakis, T. Sellis, A. U. Frank, S. Grumbach, R. H. Gueting, C. S. Jensen, N. Lorentzos, Y. Manolopoulos, E. Nardelli, B. Pernici, H. J. Schek, M. Scholl, B. Theodoulidis, and N. Tryfona, editors, *Spatio-Temporal Databases: The CHOROCHRONOS Approach*, volume 2520 of *LNCS*, pages 117–176. Springer, Berlin, 2003.
25. R. H. Güting and M. Schneider. *Moving Objects Databases*. Morgan Kaufmann Publishers, 2005.
26. I. A. R. Hulbert. GPS and its Use in Animal Telemetry: The next five Years. In A. M. Sibbald and I. J. Gordon, editors, *Proceedings of the Conference on Tracking Animals with GPS*, pages 51–60, Aberdeen, UK, 2001. Macaulay Insitute.
27. Y. Inada and K. Kawachi. Order and flexibility in the motion of fish schools. *Journal of theoretical Biology*, 214(3):371–387, 2002.
28. Y. Ishikawa, Y. Tsukamoto, and H. Kitagawa. Extracting mobility statistics from indexed spatio-temporal datasets. In *Proc. of the 2nd Workshop on Spatio-Temporal Database Management (STDBM)*, pages 9–16, 2004.
29. A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, 2003.
30. T. Kapler, R. Harper, and W. Wright. Correlating events with tracked movement in time and space: A geotime case study, 2005. Presented at the 2005 Intelligence Analysis Conference, Washington, DC.

31. G. Kollios, S. Sclaroff, and M. Betke. Motion mining: discovering spatio-temporal patterns in databases of human motion. In *Proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 2001.
32. M. Koubarakis, Y. Theodoridis, and T. Sellis. Spatio-Temporal Databases in the Years Ahead. In M. Koubarakis, T. Sellis, A. U. Frank, S. Grumbach, R. H. Gueting, C. S. Jensen, N. Lorentzos, Y. Manolopoulos, E. Nardelli, B. Pernici, H. J. Schek, M. Scholl, B. Theodoulidis, and N. Tryfona, editors, *Spatio-Temporal Databases: The CHOROCHRONOS Approach*, volume 2520 of *LNCS*, pages 345–347. Springer, Berlin, 2003.
33. J. Krause and G. D. Ruxton. *Living in Groups*. Oxford Series in Ecology and Evolution. Oxford University Press, New York, NY, 2002.
34. M. P. Kwan. Interactive geovisualization of activity-travel patterns using three dimensional geographical information systems: A methodological exploration with a large data set. *Transportation Research Part C*, 8(1–6):185–203, 2000.
35. R. F. Lachlan, L. Crooks, and K. N. Laland. Who follows whom? Shoaling preferences and social learning of foraging information in guppies. *Animal Behaviour*, 56(1):181–190, 1998.
36. P. Laube and S. Imfeld. Analyzing relative motion within groups of trackable moving point objects. In M. J. Egenhofer and D. M. Mark, editors, *Geographic Information Science 2002*, volume 2478 of *Lecture Notes in Computer Science*, pages 132–144, Berlin, 2002. Springer.
37. P. Laube, S. Imfeld, and R. Weibel. Discovering relative motion patterns in groups of moving point objects. *International Journal of Geographical Information Science*, 19(6):639–668, 2005.
38. P. Laube and R.S. Purves. An approach to evaluating motion pattern detection techniques in spatio-temporal data. *Computers, Environment and Urban Systems*, 30(3):347–374, 2006.
39. P. Laube, M. van Kreveld, and S. Imfeld. Finding REMO – detecting relative motion patterns in geospatial lifelines. In P. F. Fisher, editor, *Developments in Spatial Data Handling: Proceedings of the 11th International Symposium on Spatial Data Handling*, pages 201–214, Berlin, 2004. Springer.
40. J. B. Leca, N. Gunst, B. Thierry, and O. Petit. Distributed leadership in semifree-ranging white-faced capuchin monkeys. *Animal Behaviour*, 66:1045–1052, 2003.
41. Y. Li, J. Han, and J. Yang. Clustering moving objects. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, Seattle, WA, USA, 2004. ACM Press.
42. N. Mamoulis, H. Cao, G. Kollios, M. Hadjieleftheriou, Y. Tao, and D. Cheung. Mining, indexing, and querying historical spatiotemporal data. In *Proceedings of the 10th ACM SIGKDD International Conference On Knowledge Discovery and Data Mining*, pages 236–245. ACM, 2004.
43. D. M. Mark. Geospatial Lifelines. In *Integrating Spatial and Temporal Databases*, volume 98471. Dagstuhl Seminars, 1998.
44. H. J. Miller and J. Han. Geographic data mining and knowledge discovery: An Overview. In H. J. Miller and J. Han, editors, *Geographic data mining and knowledge discovery*, pages 3–32. Taylor & Francis, London, UK, 2001.
45. K. Mouratidis, D. Papadias, and M. Hadjieleftheriou. Conceptual partitioning: an efficient method for continuous nearest neighbor monitoring. In *Proceedings of the 2005 ACM SIGMOD Conference on Management of Data*, pages 634–645, 2005.
46. 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, London, UK, 2001.
47. R. O. Peterson, A. K. Jacobs, T. D. Drummer, L. D. Mech, and D. W. Smith. Leadership behavior in relation to dominande and reproductive status in gray wolves, *canis lupus*. *Canadian Journal of Zoology*, 80(8):1405–1412, 2002.
48. F. Porikli. Trajectory Distance Metric Using Hidden Markov Model based Representation. In *Proceedings of the 6th IEEE European Conference on Computer Vision, Workshop on PETS*, Prague, 2004.
49. Y. Qu, C. Wang, and X. S. Wang. Supporting fast search in time series for movement patterns in multiple scales. In *Seventh international conference on Information and knowledge management*, pages 251–258, Bethesda, Maryland, United States, 1998. ACM Press.
50. S. A. Rands, G. Cowlishaw, R. A. Pettifor, J. M. Rowcliffe, and R. A. Johnstone. Spontaneous emergence of leaders and followers in foraging pairs. *Nature*, 423(6938):432–434, 2003.
51. J. Raper. The Dimensions of GIScience, 2002. Keynote speech of GIScience 2002.
52. C. W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, volume 21, pages 25–34. ACM Press, 1987.

53. J. F. Roddick, K. Hornsby, and M. Spiliopoulou. An Updated Bibliography of Temporal, Spatial, and Spatio-temporal Data Mining Research. In J. F. Roddick and K. Hornsby, editors, *Temporal, spatial and spatio-temporal data mining, TSDM 2000*, volume 2007 of *Lecture Notes in Artificial Intelligence*, pages 147–163, Berlin, 2001. Springer.
54. T. Shirabe. Correlation analysis of discrete motions. In *Proceedings of the Fourth International Conference on Geographic Information Science, GIScience 2006*, volume 4197 of *Lecture Notes In Computer Science*, pages 370–382, Berlin, 2006. Springer-Verlag.
55. G. Sinha and D. M. Mark. Measuring similarity between geospatial lifelines in studies of environmental health. *Journal of Geographical Systems*, 7(1):115–136, 2005.
56. A. P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao. Modeling and Querying Moving Objects. In *13th International Conference on Data Engineering (ICDE13)*, 1997.
57. C. S. Smyth. Mining mobile trajectories. In H. J. Miller and J. Han, editors, *Geographic data mining and knowledge discovery*, pages 337–361. Taylor & Francis, London, UK, 2001.
58. J. J. Thomas and K. A. Cook. A visual analytics agenda. *IEEE Computer Graphics and Applications*, 26(1):10–13, 2006.
59. S. Tisue and U. Wilensky. NetLogo: A Simple Environment for Modeling Complexity. In *International Conference on Complex Systems*, Boston, 2004.
60. F. Verhein and S. Chawla. Mining spatio-temporal association rules, sources, sinks, stationary regions and thoroughfares in object mobility databases. In *Proceedings of the 11th International Conference on Database Systems for Advanced Applications (DASFAA)*, volume 3882 of *Lecture Notes in Computer Science*, pages 187–201. Springer, 2006.
61. U. Wilensky. NetLogo Flocking model, 1998. `http://ccl.northwestern.edu/netlogo/models/Flocking`.
62. U. Wilensky. NetLogo (and NetLogo User Manual), 1999. `http://ccl.northwestern.edu/netlogo`.
63. O. Wolfson and E. Mena. Applications of Moving Objects Databases. In Y. Manolopoulos, A. Papadopoulos, and M. Vassilakopoulos, editors, *Spatial Databases: Technologies, Techniques and Trends*. Idea group Co., 2004.
64. O. Wolfson, B. Xu, S. Chamberlain, and L. Jiang. Moving Objects Databases: Issues and Solutions. In M. Rafanelli and M. Jarke, editors, *10th International Conference on Scientific and Statistical Database Management, Proceedings, Capri, Italy, July 1-3, 1998*, pages 111–131. IEEE Computer Society, 1998.
65. X. Xiong, M. F. Mokbel, and W. G. Aref. Sea-cnn: Scalable processing of continuous k-nearest neighbor queries in spatio-temporal databases. In *Proc. of the 21st International Conference on Data Engineering (ICDE 2005)*, pages 643–654, 2005.