Decentralized Movement Pattern Detection amongst Mobile Geosensor Nodes

Patrick Laube[1], Matt Duckham[1], and Thomas Wolle[2]

[1] Department of Geomatics, The University of Melbourne, VIC 3010, Australia,
p.laube@unimelb.edu.au
[2] NICTA Sydney, Locked Bag 9013, Alexandria, NSW 1435, Australia

Abstract. Movement patterns, like *flocking* and *converging*, *leading* and *following*, are examples of high-level process knowledge derived from low-level trajectory data. Conventional techniques for the detection of movement patterns rely on centralized "omniscient" computing systems that have global access to the trajectories of mobile entities. However, in decentralized spatial information processing systems, exemplified by wireless sensor networks, individual processing units may only have access to *local* information about other individuals in their immediate spatial vicinity. Where the individuals in such decentralized systems are mobile, there is a need to be able to detect movement patterns using collaboration between individuals, each of which possess only partial knowledge of the global system state. This paper presents an algorithm for decentralized detection of the movement pattern *flock*, with applications to mobile wireless sensor networks. The algorithm's reliability is evaluated through testing on simulated trajectories emerging from unconstrained random movement and correlated random walk.

1 Introduction

Movement patterns represent high-level process knowledge derived from low-level trajectory data [1]. They are the spatiotemporal "trace" left behind by the behavior of moving entities [2]. Examples of movement patterns include *flocking* as in a "mob" of sheep [3], *leading* and *following* found in group dynamics [4, 5], or *converging* and *diverging* of pedestrians in crowding scenarios [6, 7]. Figure 1 illustrates the movement pattern of a prototypical "flock". The figure shows that at a particular time instant t, four moving entities a, b, d, e are in close spatial proximity (all lie within a circle of radius p). A commonsense interpretation of a flock is where mobile individuals maintain such spatial proximity over an extended period of time.

Wireless sensor networks (WSN) are increasingly applied in a dynamic context. So-called *mobile wireless sensor networks* (mWSN) provide new opportunities for monitoring and understanding coordinated movement processes, with a wide range of applications including "smart" farming [8], emergency response [9], robotics [10], and in-car navigation [11]. Conventional techniques for the detection of movement patterns rely on centralized "omniscient" computing

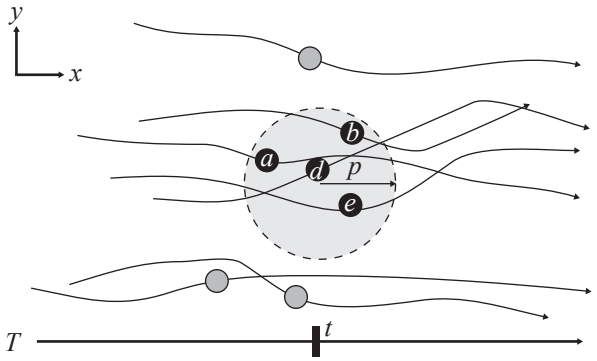

Fig. 1. A "flock" can be defined as a set of entities moving in spatial proximity for some specified time period.

databases and systems that have global access to the trajectories of mobile entities. However, in decentralized spatial information processing systems, like mWSNs, individual processing units may only have access to *local* information about the movement of other individuals in their immediate spatial vicinity. This paper presents a decentralized algorithm that enables moving sensor nodes in an mWSN to infer if they are part of an ongoing flocking movement pattern. The algorithm uses only local collaboration with no central control, and relies on qualitative spatial information (i.e., it does not require precise coordinate information about the location of individuals). Like most tractable centralized flock detection algorithms, our decentralized algorithm generates approximate solutions, so the paper includes an empirical analysis of reliability of the algorithm.

The remainder of this paper is structured as follows. Section 2 reviews the relevant literature on mining movement patterns and mobile wireless sensor networks. Section 3 introduces the underlying model of mobility and flocking in an mWSN. Section 4 presents the algorithm for detecting flocks in an mWSN, termed FLAGS. A series of simulation experiments are used to evaluate the reliability and efficiency of FLAGS in Sect. 5. The paper discusses the findings of this work in Sect. 6 and concludes with final remarks and an outlook in Sect. 7.

2 Background

2.1 Mining Movement Patterns

Various data mining techniques have been used to detect generic movement patterns [2]. In 2004, Laube et al. [12] defined a collection of spatiotemporal patterns based on direction of movement and location, e.g. *flock*, *leadership*, *convergence* and *encounter*, and they gave algorithms to compute them efficiently. Several

subsequent articles improved the formalization and the algorithmic discovery of such patterns.

Benkert et al. [13] modified the original definition of a flock to be a set of entities moving close together during a time interval. The applied data mining approach bases on projection of 2D trajectories into multidimensional space and query operations on quadtrees. This approach results in efficient approximation algorithms for finding such flock patterns of a fixed length and a fixed number of flock entities, where the radius is approximated within a factor of 2. For the same definition of flock, Gudmundsson and van Kreveld [14] showed that for any radius approximation with factor smaller than 2, computing the longest duration flock and the largest subset flock is NP-hard to compute and even NP-hard to approximate within a factor of $|T|^{1-\epsilon}$ and $|A|^{1-\epsilon}$, respectively, where T is the set of all discrete time steps, A is the set of all sensor nodes, and $\epsilon > 0$. As another such movement pattern, Andersson et al. [15] gave a generic definition of the pattern *leadership* and discussed how such leadership patterns can be computed from a group of moving entities. This work revolves around the concept of leading as "being followed but not following anyone else" for some time. Mining for such leadership patterns involves a set of algorithms operating on a set of globally preprocessed data arrays that store leading and following constellations for the investigated set of entities. All such movement patterns have in common that some geometrical relation of the involved moving objects has to persist over a certain time span.

Current data mining approaches for movement patterns rely on global knowledge and global data structures such as quadtrees, clustering, or preprocessed metadata arrays. The use of such global and hence static data structures is not well-suited for local knowledge discovery in a dynamic scenario such as a mobile wireless sensor network.

2.2 Mobile Wireless Sensor Networks

A *wireless sensor network* (WSN) is a wireless network of untethered, battery powered miniaturized computers with the ability to sense, process, and communicate information in a collaborative way [16]. A WSN monitoring a phenomenon in geographic space is called *geosensor network* [17]. The tracking of mobile targets is one typical task for geosensor networks [16, 18]. Other applications have included observing hazards [19], monitoring seismic activity [20, 21], or managing traffic flow [11]. When the sensor nodes are deployed to moving entities in a dynamic scenario, we refer to a *mobile wireless sensor network* (mWSN).

Mobility presents significant challenges to the design of mWSN applications. Maintaining static, global data structures, like connectivity graphs and routing tables, is a particular challenge in mWSN. Even in static WSN, battery power resources for wireless communication are extremely limited. With mWSN, constantly changing network topologies and sensed data means maintaining centralized global knowledge and data structures can become highly complex and inefficient.

As a result, some researchers have turned to *decentralized algorithms* (an algorithm that runs in parallel on sensor nodes without any centralized control) to address this problem. Unlike global approaches, decentralized algorithms facilitate fast local updating and do not require hard-to-maintain global consistency [22]. Other researchers including [19, 23, 24, 25] have all investigated decentralized knowledge maintenance and creation in spatial applications to cope with the problems posed by mobility in mWSN. Such decentralized approaches can also help to increase energy efficiency [16], increase scalability and robustness [26], and reduce the potential for information overload [27, 28].

As well as challenges, mobility also presents opportunities that are beginning to be exploited by some researchers. Several authors exploit the *mobility diffusion effect*, that is the diffusion of information through an mWSN from the constant hand-over of information amongst meeting moving nodes [29]. *Last encounter routing* (LER), for example, computes routes purely based on a last encounter table stored in every node. The constant rearrangement of nodes in an mWSN can furthermore be exploited to overcome unfavorable constellations. Grossglauser and Tse [22], for example, showed that for asynchronous applications with high delay tolerance, patience can increase efficiency with respect to throughput capacity when messages can wait for good routes in a network topology constantly changing due to node mobility.

In summary, the constantly changing topology amongst the moving nodes in an mWSN challenges conventional solutions that have been developed for static networks, but at the same time offers new options for in-network data processing through the exploitation of the spatiotemporal nature of movement. Decentralized spatial computing is well-suited to use in mWSN, capable of operating using purely *local* knowledge, but still aiming to monitor geographic phenomena with *global* extents [30].

3 Problem Definition

In this section, we provide a formal problem definition, including specifying the assumptions behind decentralized processing in an mWSN, and providing a precise definition of the meaning of "flock" in our scenario.

3.1 Preliminaries and Assumptions

An mWSN can be modeled as a set A of sensor nodes. The locations of sensor nodes are known for an ordered set of discrete time steps $T = \{t_1, t_2, ..., t_n\}$. The associated movement trajectories of sensor nodes in the plane ($\mathbb{R}^2$) can be modeled as a *locator* function $l : A \times T \rightarrow \mathbb{R}^2$. Thus for any time $t \in T$ and sensor node $a \in A$, $l(a, t)$ gives the coordinate location of sensor node a at time t. The distance between two sensor nodes $a, a' \in A$ at time $t \in T$ is given by the usual Euclidean metric $\delta(l(a, t), l(a', t))$. In the sequel, we write $\delta_t(a, a')$ to denote the distance between two sensor nodes a and a' at time t for conciseness, i.e., $\delta_t(a, a') := \delta(l(a, t), l(a', t))$.

At this point a few assumptions are worth noting. Without loss of generality we make the simplifying assumption that the set of sensor nodes is constant over time (i.e., that no sensor nodes leave or enter the network). We model time as a discrete domain T. This assumption also does not lead to a loss of generality, since continuous time can always be adequately approximated in a specific application with arbitrarily fine discrete time steps.

Nearby sensor nodes in an mWSN can wirelessly communicate with one another. Thus at any given point in time, each sensor node will have a (possibly empty) set of sensor nodes within its communication range, called its *neighborhood*. Given a fixed communication distance $c \in \mathbb{R}^+$, the neighborhood of a sensor node a at time t, written $nbr(a,t)$, is the set of sensor nodes within a's communication distance at time t, i.e., $nbr(a,t) := \{a' \in A | \delta_t(a,a') \leq c\}$. Note that this model assumes all sensor nodes have constant and equal communication distances. Although this is not realistic in actual sensor node networks, it is adequate for the purposes of this paper and helps simplify the formal model.

One further point worth noting is that although the locator function provides the location of each sensor node in the plane over time, our algorithm does *not* assume that individual sensor nodes have access to this information. Instead the algorithm in the next section is purely qualitative, relying instead on the neighborhoods of each sensor node. These neighborhoods emerge as a consequence of the physical characteristics of wireless communication, without the need for quantitative positioning systems such as GPS or range-finding.

3.2 Decentralized Detection of Flocks

Laube et al. [12] use the term "flocking" to describe a collective movement pattern expressed by a set of moving entities. In our context, the entities refer to the moving sensor nodes of a mWSN. A flock in this sense is any set M of n mobile sensor nodes that exhibit the same motion attribute over a given period of time of length $k \in \mathbb{R}$. Speed or movement azimuth are examples of motion attributes in this context. This initial definition is very general and does not assume any notion of proximity, which is often associated with the term flocking in common usage.

Adding a proximity constraint, [13] defined an (n,k,p)-*flock* as any set M of n mobile sensor nodes, where for every moment in a time period of k consecutive time steps, there exists some disk of pattern radius p that contains every sensor node in M. More formally and in the context of our definition of mWSN above, an (n,k,p)-*flock* is a set $M \subseteq A$ such that for every time instant $t \in \{t_i, ..., t_j\} \subseteq T$, with $j - i + 1 \geq k$, there exists a circle of radius p that spatially contains $l(m,t)$ for all $m \in M$.

The definition of an (n,k,p)-*flock* thus assumes a flock exists for a period of k consecutive time steps single. Further, the definition assumes that the flock is composed of n identical sensor nodes for its entire existence. As such, the definition is quite restrictive. Current work by the authors is looking at relaxing these constraints, and discussion of the options has already appeared in the literature [3].

The problem we address in this paper is a decentralized detection of (n, k, p)-*flocks*. Given a set of sensor nodes A, we give an algorithm that detects for any time t all present (n, k, p)-*flocks* by only relying on *local* knowledge and without any form of global information processing. As our main objective is to substitute global approaches with a purely decentralized approach, we primarily investigate the reliability of our inherently approximating decentralized solution. In this context, reliability refers to an error analysis, quantifying the number of correctly detected and missed patterns, respectively.

4 The FLAGS Algorithm

As part of this work we have developed a family of decentralized algorithms for flock detection. This section presents the FLAGS algorithm for detecting **fl**ocking **a**mongst **g**eo-**s**ensors in an mWSN. FLAGS is designed to find flock patterns without central control, solely by decentralized collaboration of roaming sensor nodes, that only perceive their immediate neighbors via simple "hello" messages.

The algorithm is distributed as it runs in parallel on all sensor nodes at the same time. The algorithm is decentralized in the sense that each sensor node a can only access information about its immediate neighborhood, specifically the identity of sensor nodes that are within direct one-hop communication range of a at time t, i.e., $nbr(a, t)$. The algorithm is dynamic, because each sensor node is constructing new information about the existence of flocks on-the-fly as it moves through time and space.

4.1 Handing around Maturing Information Tokens

In the FLAGS algorithm, the collective of roaming sensor nodes administers information tokens that accumulate knowledge about flock patterns over time. A token is simply a pair (X, j), where X is a set of sensor nodes that a knows currently lie within a circle of radius p during the previous j time steps. The set of tokens stored by a sensor node a at time t is denoted $\mathit{TokenSet}(a, t)$.

At each time step t_i the algorithm works as follows (see Algorithm 1): Each sensor node a computes its set of neighbors $nbr(a, t_i)$ by sending and receiving simple "hello" messages (line 1). If as current set of neighbors contains at least as many sensor nodes as are required for detecting a flock (denoted by the threshold ν, related to the number of nodes in the flock n and discussed further in the next section), a will create a new token containing this information and add it to $\mathit{TokenSet}(a, t_i)$ (lines 3–4). Initially, $\mathit{TokenSet}(a, t_i)$ is empty (line 2). Next, a will update and check all its tokens $(X, j) \in \mathit{TokenSet}(a, t_{i-1})$ from the previous time step t_{i-1} (lines 5). The token (X, j) is updated by $X := X \cap nbr(a, t_i)$ to indicate that now only a smaller set of sensor nodes are in a's neighborhood, and by $j := j + 1$ to reflect the increased number of time steps that the sensor nodes in the updated X have been neighbors of a (line 6). If the size of the updated X is at least the number of sensor nodes required for detecting a flock ν, the token is added to $\mathit{TokenSet}(a, t_i)$ (lines 7–8). After that, the sensor node a

inspects all tokens in $TokenSet(a, t_i)$, and whenever it finds a token of age $j \geq k$ it triggers a "pattern found" message (lines 9–11). As a last but very important step, sensor node a will exchange its set of tokens with its neighbors (line 12). Figure 2 illustrates this procedure and Algorithm 1 formalizes it.

FLAGS decouples knowledge about patterns from sensor nodes. Knowledge diffuses through the network in form of knowledge collecting tokens that are handed over between roaming sensor nodes. Even if all members of a flock rearrange at each consecutive step, the spatiotemporal knowledge describing that flock is very likely to persist, because it is likely that at least one sensor node holds an appropriate token.

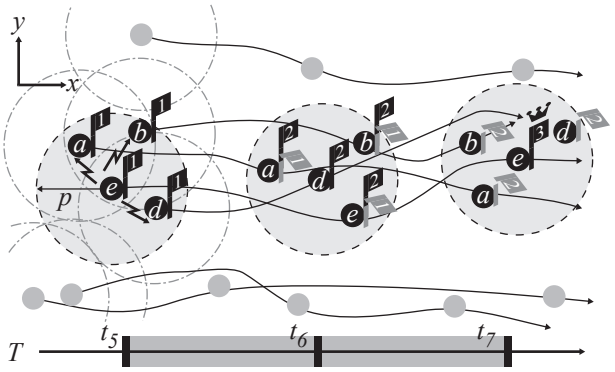

Fig. 2. Maturing knowledge tokens detect a $(n = 4, k = 3, p)$-*flock* locally when $c \approx p$. At t_5 sensor node e counts the required 4 neighbors, creates a token $(\{a, b, d, e\}, 1)$ that is transmitted to all its neighbors within communication range c (little numbered flags). At t_6, after having moved and potentially rearranged, all sensor nodes check their tokens. This time only sensor node d counts enough neighbors and ages his token to $(\{a, b, d, e\}, 2)$. All other sensor nodes drop their token. Sensor node d, however, forwards its aged token to all its neighbors. Finally, at t_7, again e counts enough neighbors, its token $(\{a, b, d, e\}, 3)$ reaches the mature age $k = 3$, and flags a "found pattern"-message (crown).

4.2 Local Extrapolation

The description of the FLAGS algorithm in the previous section was deliberately vague about the precise value of ν, the "number of sensor nodes required for detecting a flock." This section explains how a correct value for ν is chosen in the FLAGS algorithm.

As discussed above, the spatial awareness of the roaming sensor nodes is constrained by their neighborhood, which in turn is constrained by their communication range c. As a result, the ratio of the sensor node's communication

Algorithm 1: Procedure that is locally run once at each time step t_i in each sensor node a for a global value ν.

```
// initialization
communicate "hello" messages and construct set of neighbors nbr(a, t_i)
TokenSet(a, t_i) ← ∅
// create new token if the neighborhood is large enough
if |nbr(a, t_i)| ≥ ν then
    TokenSet(a, t_i) ← TokenSet(a, t_i) ∪ {(nbr(a, t_i), 1)}
// check tokens of previous time step; if valid, update and age them
foreach (X, j) ∈ TokenSet(a, t_{i−1}) do
    X_new ← X ∩ nbr(a, t_i)
    if |X_new| ≥ ν then
        TokenSet(a, t_i) ← TokenSet(a, t_i) ∪ {(X_new, j + 1)}
// check for patterns in current set of tokens
foreach (X, j) ∈ TokenSet(a, t_i) do
    if j ≥ k then
        report "pattern found"
// information diffusion
communicate with neighbors and update TokenSet(a, t_i)
```

range c and the pattern radius p, informally termed a node's *perception range*, plays a critical role in detecting flocks in a decentralized algorithm. Figure 3 illustrates the $\frac{c}{p}$-ratio for $c \ll p$, $c \approx p$, and $c \gg p$, respectively.

Constellations $c \gg p$ are of limited interest from a decentralized spatial computing perspective, since in these cases there is every chance a single sensor node will locally be able to observe the entire flock (ultimately converging toward sensor nodes having global knowledge). By contrast, $c \approx p$, and particularly $c \ll p$ require strategies and heuristics allowing sensor nodes to overcome their own limited perception range.

To correct for limited communication range, FLAGS applies a local extrapolation heuristic, by assuming that a sensor node that is part of a flock can expect a neighborhood size that is in proportion to its perception range. In other words, a sensor node for which $c \ll p$ detects a flock constellation when its neighborhood contains approximately $\nu = \frac{c^2}{p^2} * n$ sensor nodes. In Fig. 3(a) the central sensor node has a neighborhood of 5 (including itself) and hence might infer the presence of a flock of size $n = 20$ having sensed the required fraction (in this case: $\frac{1}{4}$) of n.

However, this first approximation fails as it assumes that some sensor node is located at the exact center of the flock. Any eccentricity in the most central node's location means a sensor node's communication area is expected to only cover a smaller part of the flock area, shown in Fig. 3(d). As a result, we expect a sensor node in a flock to sense on average slightly fewer than $\frac{c^2}{p^2} * n$ neighbors due to its eccentricity with respect to the center of the flock.

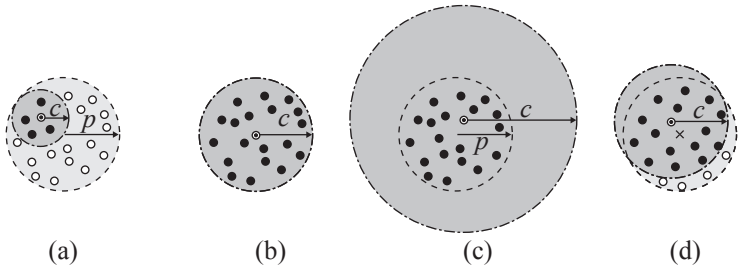

Fig. 3. The $\frac{c}{p}$-ratio rules the difficulty of detecting flock patterns in a decentralized way, illustrated are the ratios $c = 0.5p$ (a), $c = 1p$ (b), and $c = 2p$ (c) respectively. Furthermore, even if $c = 1p$ there is no guarantee for "seeing" all sensor nodes of a flock, as the most central sensor node may be eccentric (d).

To correct for this effect a compensation factor E was introduced. E was computed as the expected fraction of the flock area (a circle of radius p) that is covered by the communication area (a circle of radius c, with center inside the flock area) of a uniformly randomly placed sensor node inside this flock. The formal mathematical details of the derivation of E are outside the scope of this paper. Instead we provide a short explanation of our approach in the following paragraph and an illustrative Fig. 4 plotting E over various $\frac{c}{p}$ ratios .

To compute E, we model the flock and communication areas as two disks in the plane with radius p and c, respectively. The distance between the centers of the disks is denoted by s. These disks have nonempty intersection, since $s \leq p$. We computed the area of intersection as a function A of p, c and s. We then defined a function R of p, c and s, as the ratio of the area A and the flock area. The expected value of this function for a uniformly randomly chosen s with $0 \leq s \leq p$, is derived from the integral of R from $s = 0$ to $s = p$. This yields a function E of p and c. But since R and E are defined to be ratios, E is only a function of $\frac{c}{p}$ and not the actual values of p and c.

In the FLAGS algorithm, the threshold ν is used as the number of sensor nodes required for detecting a flock, defined as follows:

$$\nu := E\left(\frac{c}{p}\right) \cdot n$$

The effect of the compensation factor E is to reduce the number of neighbors a sensor node that is part of a flock expects to have. As discussed above, this reduction allows for the likelihood that, on average, sensor nodes are not ideally situated to perceive the best possible number of neighbors.

5 Evaluation

This section reports on implementation and testing of the FLAGS algorithm. In addition to testing the reliability of the algorithm, this section also investigates

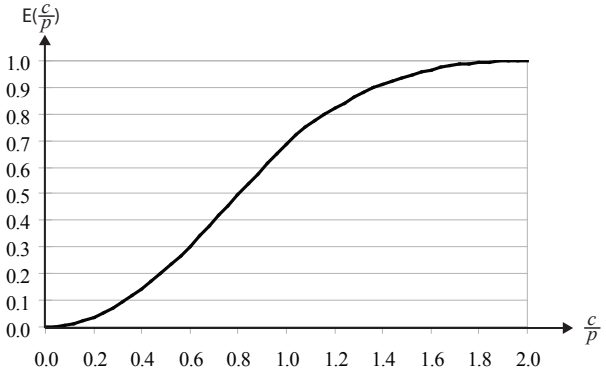

Fig. 4. Compensation factor E over $\frac{c}{p}$. For $c = 0.5p$ we get $E(0.5) = 0.22$, for $c = 1p$ the compensation factor is $E(1.0) = 0.69$, and for $c = 2p$ it is $E(2.0) = 1.0$.

to what degree the underlying movement regime influences the performance of our decentralized knowledge discovery algorithm. For evaluation purposes, a simulation environment for detecting flocks in given trajectories was implemented using the popular open source agent-based modeling toolkit REPAST (see Fig. 5).

5.1 Data

Simulated trajectory data was used instead of real movement observation data, as this allows control of both the movement regime and the flocking behavior of the moving sensor nodes in the experiments. Different movement regimes were expected to have a significant influence on the performance of the FLAGS algorithm, so simulated trajectories that follow well-known movement regimes were used: unconstrained random movement (URM) and correlated random walk (CRW). Using simulated movement observation data also had the experimental advantage that the number of flocks present in the data could be easily controlled. This was crucial for the performance analysis, which evaluates how many of the implanted patterns are actually found by the FLAGS algorithm.

For all experiments a population A of sensor nodes ($|A| = 200$) was generated moving over $|T| = 100$ discrete time steps in an underlying square space of size $8192 * 8192$ positions. 50 out of 200 sensor nodes were given flocking behavior, implanted in five flocks consisting of $n = 10$ flocking sensor nodes each with a pattern radius $p = 250$. The flocking behavior follows the definition of *flock* outlined in Sect. 3.2.

Unconstrained Random Movement (URM). In order to maximize the comparability of our decentralized approach with previous work, the first set of experiments used the same unconstrained random movement (URM) generator as [13]. In the URM data sets, the location of a sensor node is completely random,

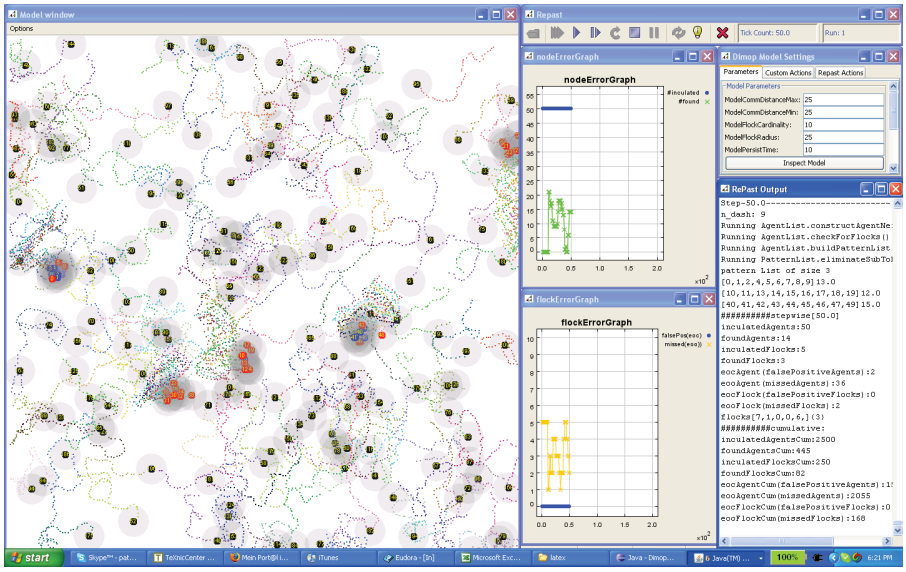

Fig. 5. FLAGS-implementation in REPAST. The framework features a map view (communication ranges as gray shades, trajectories as dotted lines), two error plots (*eoo*, *eoc*), a parameter and a log window.

and unrelated to any previous steps. Non-flocking sensor nodes randomly "jump" around in simulation space. Flocking sensor nodes, however, are at each time step randomly positioned withing a circle of size pattern radius $p = 250$, itself randomly positioned.

Correlated Random Walk (CRW). Initial experiments showed that URM only allowed for very limited exploitation of the spatiotemporal characteristics of movement. Anticipating that any decentralized pattern detection approach would rely in part on exactly such exploitation, a second set of experiments was conducted with trajectories generated using correlated random walks.

In a random walk, the location of a sensor node in successive timesteps is determined by a randomized displacement from the previous time step. In correlated random walk (CRW) steps of sensor nodes are correlated by making the direction and/or step length of the current move bias the direction and step length of the next move [31]. Step length and turning angle of the subsequent move are typically drawn from some stochastic frequency distribution, for instance in the case of direction with turning angles concentrated around $\mu = 0$. CRW provides a more realistic model of random spatial movement than URM, as consecutive locations of sensor nodes are highly correlated in space.

In the experiments a normal distribution with a direction change $N(\mu = 0, \sigma = 0.6 * \pi)$ and a step length of $N(\mu = 50, \sigma = 15)$ was used. Each sensor

node walked a trajectory of approximate length $100*50 = 5000$ units. In addition to the CRW property, flocking sensor nodes satisfied the flock property, by simply ensuring that they were aggregated within the given pattern radius $p = 250$. The map view in Fig. 5 illustrates first 50 time steps of a CRW data set.

5.2 Experiments

Following the general WSN constraint that communication consumes more energy than sensing, algorithm efficiency often bases on keeping the number of messages low. For the evaluation of the here discussed decentralized data mining application, a slightly different perspective was taken. It was assumed that our sensor nodes constantly know in a qualitative manner by which neighbors they are surrounded. Since it had to be anticipated that the decentralized data mining algorithm is rather an approximation than an exact solution, performance was evaluated as the ratio of patterns found with the centralized and the decentralized approach. For the sake of simplicity, this evaluation focused on the pattern detection and did not investigate the details of routing the information back to the query source once the patterns had been detected.

FLAGS reliability was tested for both URM data and CRW data (Fig. 6). In both cases two levels of flock contiguity were tested: In order to be detected flocks were required to be detected for $k = 3$, and $k = 10$ consecutive time steps out of $|T| = 100$ time steps in total. Note that each flock could only be counted once, even if several sensor nodes would find the same flock. For a total of 5 simulation runs and four combinations each, the *detection error* for a variable $\frac{c}{p} - ratio$ was computed.

- *detection error* (y-axis): At each time step the number of found patterns was compared with the known number of implanted patterns. Errors of omission (*eoo*) and errors of commission (*eoc*) were evaluated and averaged over all $|T| = 100$ time steps[3].
  - *eoo* ("missed patterns"): As 5 flocks consisting of 10 sensor nodes each were implanted, over $|T| = 100$ time steps a total 500 flocks could be detected correctly. An experiment run missing 100 flocks would result in $eoo = 0.2$.
  - *eoc* ("false positives"): FLAGS uses heuristics in order to infer the presence of patterns (local extrapolation and compensation factor E, see Sect. 4.2). Hence, FLAGS may detect patterns where there are no patterns to be found. Such false positives could, for example, emerge when $c \ll p$ and $\nu = 2$, as it might happen that two independent sensor nodes randomly intermingle for a long enough time. All such false positives were counted and again averaged over the whole experiment run. Again, with a total of 500 flocks per run, a total of 50 false positives would result in $eoc = 0.1$.

[3] Please note that in fact the error was averaged over $|T| - k + 1$ time steps, as in the first $k - 1$ time steps no flocks can be detected due to their temporal extent.

– $\frac{c}{p} - ratio$ (x-axis): FLAGS performance was evaluated for a variable $\frac{c}{p} - ratio$. Given a pattern radius $p = 250$ the communication range c was varied in the interval of $[0, \ldots, 500]$, hence in a $\frac{c}{p} - ratio$ interval of $[0, \ldots, 2]$.

Obviously, the above evaluation experiments do not account for flocks that may randomly emerge in the generated data set, but were not purposely implanted in the first place. However, the emergence of such random flocks is minimized by a relatively large flock size $n = 10$, and gets very unlikely with increasing flock contiguity k. Furthermore, such random flocks would only influence *eoc*, as in fact existing random flocks could be misinterpreted as false positives. Hence, our evaluation, in the worst case, overestimates *eoc*.

5.3 Results

All four graphs show a few similar properties (Fig. 6). No results are provided when $\frac{c}{p} < 0.48$, as below that threshold ν drops below 2 and hence pattern detection is not possible anymore (see Fig. 4). Furthermore, the error graphs often show "zigzagging" shape. This feature emerges since the number of counted neighbors inherently changes in discrete steps and hence FLAGS performance alters in grades.

URM, $k = 3$. As long as the communication range c is slightly larger than the pattern radius p, no patterns are missed out and *eoc* is 0. Around $\frac{c}{p} \approx 1.0$ a first *eoo*-peak emerges when large numbers ν are required for token generation. Below 1.0, the performance decreases gradually, as flock sensor nodes randomly shuffled in the flock disk result in more and more misses. No *eoc* was recorded. Given that the sensor nodes relocate randomly at each step, it is very unlikely that a significant number of sensor nodes satisfy the flock definition randomly.

URM, $k = 10$. As with $k = 3$, *eoo* emerges at $\frac{c}{p} < 1.2$. However, with $k = 10$ it is almost impossible that any sensor node re-finds the very *same* neighbors out of the $n = 10$ neighbors, hence FLAGS' performance degrades rapidly to total failure shortly below 0.8. Again, and for the very same reasons as above, no *eoc* was recorded.

CRW, $k = 3$. In the CRW experiment FLAGS performs significantly better than with URM, for $k = 3$ the performance is almost flawless. A t-test for 5 simulation runs indicates that for any ratio $\frac{c}{p} < 0.84$, *eoo* for CRW is significantly lower than for URM on the 95% significance levels. The explanation for this good performance lies in the locality property of CRW. Even when c drops below p, subgroups within a flock tend to stay together and hence knowledge about subgroups persists, allowing the consistent extrapolation of pattern knowledge using the compensation factor $E(\frac{c}{p})$. Also *eoc* shows moderate levels and only expresses noticeable values when ν reaches 2 at $\frac{c}{p} = 0.56$. The random event of two nodes staying together for $k = 3$ time steps is moderately likely with the chosen parameters and this *eoc* peak is hence unsurprising.

CRW, $k = 10$. Again, *eoo* is moderate, but admittedly on a higher level (according to a t-test significantly higher at the 95% level for $0.78 < \frac{c}{p} < 1.12$). The constraint of $k = 10$ makes it harder to maintain knowledge about subgroups. On a positive note, the *eoc* does not peak at all as it is just too unlikely that random patterns emerge with even $\nu = 2$.

6 Discussion

Three major findings emerge from our experiments. First, with the used parameters, *eoc* is not an issue. Apart from the single peak with small c for $k = 3$, *eoc* is negligible. This can be explained that with the used parameters even for small νs and short ks it is very unlikely that enough sensor nodes stay together for enough time steps. Different flock parameters and different densities amongst non-flocking sensor nodes may admittedly result in higher *eoc* values. Second, longer contiguities k are understandably harder to detect as a longer time required for tokens to mature opens up more possibilities for failure. Third, FLAGS performs much better for CRW than for URM, as we would expect from an algorithm developed for detecting structure in structured movement data. Since FLAGS collects neighborhood knowledge over several consecutive time steps, locality where individuals tend to stay together for some time, obviously helps for the crucial token maturing. Even though we assumed in our experiments random movement, we could show that FLAGS performs better as movement becomes less random. We hence argue that the FLAGS algorithm is taking advantage of the opportunity of movement rather than purely dealing with the challenges of movement.

This last finding agrees with similar experiences in the LER experiments in [29]. In their experiments the performance of their history based routing algorithm depended heavily on the nature of the mobility regime and the parameterization of the experimental setup. LER succeeds only since the sensor nodes perform random walks and thus sensor node mobility diffuses estimates of the destination's location sufficiently quickly and densely throughout the dynamic mWSN. Even though the preferred regime was random walk in their case and correlated random walk in ours, it seems apparent that the inevitably very specialized algorithms for decentralized spatial computing operate best within the constraints that they have been designed for.

Our initial experiments with decentralized data mining suggest that we simply have to accept the fact that any decentralized solutions inferring global knowledge from partial and potentially imperfect local information must allow for approximation solutions. One way of giving the limited sensor nodes the leeway to do so is the use of heuristics, as applied with the local extrapolation approach and the compensation factors E in FLAGS. We allow our sensor nodes to detect and report patterns even if they actually only found fractions of patterns. For many decentralized data mining applications we will have to allow for some error in order to achieve a workable solution at all. For the parameterizations used in our experiments, the heuristics provided very useful tools

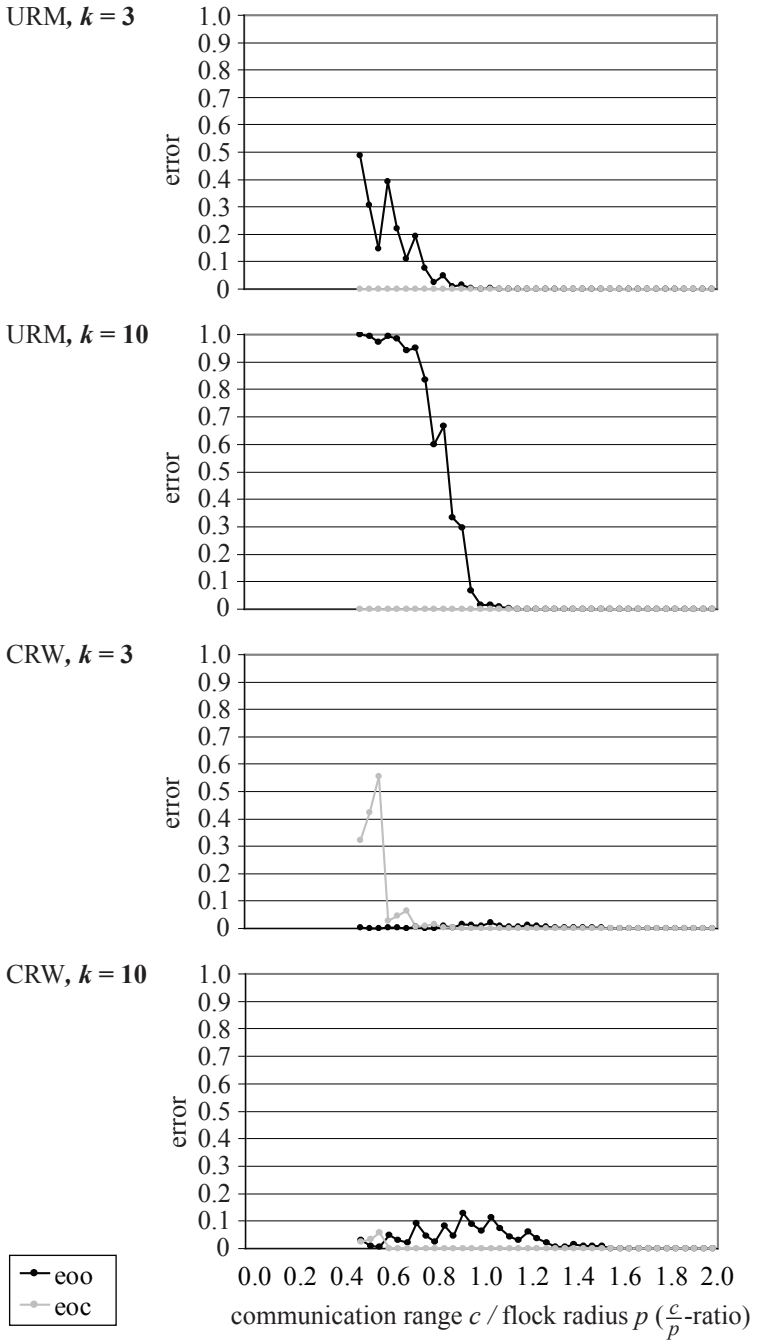

Fig. 6. Error analysis for URM and CRW experiments.

without introducing too much of *eoc*. In other cases one might have to accept some *eoc* in order to keep *eoo* low. Obviously, decentralized approximation solutions could always be used as a cost-effective preliminary data mining step that might trigger more reliable but also more expensive approaches.

FLAGS exploits the spatiotemporal properties of movement as it features information exchange amongst roaming neighbors. This constant process of exchanging and validating information allows individual sensor nodes to learn from their neighbors about processes beyond their own limited perception range. However, recording of the past and hence "memory" is purely outsourced to the knowledge tokens, the sensor nodes themselves are oblivious. Clearly, alternative approaches can be thought off where individual sensor nodes, other than exchange their knowledge with their neighbors, also record what they see along their trajectory, and integrate such spatiotemporal knowledge in order to gain the big picture beyond their limited perception range.

As mentioned earlier, we use the same definition of flock and the same data generator (for URM trajectories) as in [13]. However, a direct comparison between the two studies is not meaningful, because in [13], the results focus on theoretical and experimental running times, while the evaluation of our approach bases on an error analysis, investigating on the reliability of a decentralized solution.

7 Conclusions

In this paper we revisited the generic movement pattern "flock" and showed that such patterns formed by individuals are very suitable for decentralized and hence "ego-centric" pattern detection algorithms. We introduced FLAGS (**fl**ocking **a**mongst **g**eo-**s**ensors) a solely decentralized flock detection algorithm. FLAGS exploits the spatiotemporal properties of movement as roaming sensor nodes exchange information tokens that collect knowledge about patterns. Experiments with simulated movement data show that FLAGS successfully completes its task without central control when *a priori* knowledge about the movement regime can be exploited.

As a first general lesson learned, we argue that separating knowledge from sensor nodes (in our specific case in the form of "floating knowledge tokens"), is a promising strategy for overcoming the limited spatial perception of individual sensor nodes in an mWSN. Second, we acknowledge that in a decentralized setting heuristics are a suitable way for compensating for the limited perception of individual sensor nodes. We thirdly identify the need to further explore locality. Hence, currently we are working on improved versions of FLAGS that use a memory function. Sensor nodes with memory are enabled to "graze" information in space-time and hopefully assemble spatially limited glimpses in order to form the bigger picture.

References

[1] Galton, A.: Dynamic collectives and their collective dynamics. In Cohn, A., Mark, D.M., eds.: Spatial Information Theory, Proceedings. Volume 3693 of Lecture Notes in Computer Science. Springer-Verlag, Berlin (2005) 300–315

[2] Gudmundsson, J., Laube, P., Wolle, T.: Movement patterns in spatio-temporal data. In Shekhar, S., Xiong, H., eds.: Encyclopedia of GIS. Springer, Berlin Heidelberg (in press)

[3] Gudmundsson, J., van Kreveld, M., Speckmann, B.: Efficient detection of patterns in 2D trajectories of moving points. GeoInformatica **11**(2) (2007) 195–215

[4] Andersson, M., Gudmundsson, J., Laube, P., Wolle, T.: Reporting leaders and followers among trajectories of moving point objects. GeoInformatica (in press)

[5] Shirabe, T.: Correlation analysis of discrete motions. In: Geographic Information Science, Proceedings. Volume 4197 of Lecture Notes in Computer Science. Springer-Verlag, Berlin (2006) 370–382

[6] Laube, P., Imfeld, S., Weibel, R.: Discovering relative motion patterns in groups of moving point objects. International Journal of Geographical Information Science **19**(6) (2005) 639–668

[7] Batty, M., Desyllas, J., Duxbury, E.: The discrete dynamics of small-scale spatial events: agent-based models of mobility in carnivals and street parades. International Journal of Geographical Information Science **17**(7) (2003) 673–697

[8] Wark, T., Corke, P., Sikka, P., Klingbeil, L., Guo, Y., Crossman, C., Valencia, P., Swain, D., Bishop-Hurley, G.: Transforming agriculture through pervasive wireless sensor networks. Pervasive Computing, IEEE **6**(2) (2007) 50–57

[9] Russell, M.: Attention all units: Dick tracy is on its way. The Sunday Age (2007) 9

[10] Correll, N., Martinoli, A.: Collective inspection of regular structures using a swarm of miniature robots. In Ang, J., Marcelo, H., Khatib, O., eds.: Experimental Robotics IX, The 9th International Symposium on Experimental Robotics (ISER), Singapore, June 18-21. Springer Tracts in Advanced Robotics. Springer

[11] Kellerer, W., Bettstetter, C., Schwingenschlogl, C., Sties, P., Steinberg, K.E.: (auto) mobile communication in a heterogeneous and converged world. IEEE Personal Communications **8**(6) (2001) 41–47

[12] Laube, P., van Kreveld, M., Imfeld, S.: Finding remo - detecting relative motion patterns in geospatial lifelines. In Fisher, P.F., ed.: Developments in Spatial Data Handling. Proceedings of the 11th International Symposium on Spatial Data Handling. Springer, Berlin Heidelberg, DE (2004) 201–214

[13] Benkert, M., Gudmundsson, J., Hübner, F., Wolle, T.: Reporting flock patterns. In: Algorithms - ESA 2006, Proceedings. Volume 4168 of Lecture Notes in Computer Science. Springer-Verlag Berlin, Berlin (2006) 660–671

[14] Gudmundsson, J., Van Kreveld, M.: Computing longest duration flocks in spatio-temporal data (November 1011, 2006 2006)

[15] Andersson, M., Gudmundsson, J., Laube, P., Wolle, T.: Reporting leadership patterns among trajectories. In: 22th Annual ACM Symposium on Applied Computing, Seoul, Korea (2007) 3 – 7

[16] Zhao, F., Guibas, L.J.: Wireless Sensor Networks – An Information Processing Approach. Morgan Kaufmann Publishers, San Francisco, CA (2004)

[17] Nittel, S., Stefanidis, A., Cruz, I., Egenhofer, M.J., Goldin, D., Howard, A., Labrinidis, A., Madden, S., Voisard, A., Worboys, M.: Report from the first workshop on geo sensor networks. ACM SIGMOD Record **33**(1) (2004)

[18] Guibas, L.J.: Sensing, tracking and reasoning with relations. Signal Processing Magazine, IEEE **19**(2) (2002) 73–85
[19] Duckham, M., Nittel, S., Worboys, M.: Monitoring dynamic spatial fields using responsive geosensor networks. In: 13th annual ACM international workshop on Geographic Information Systems, Bremen, Germany, ACM Press (2005) 51–60
[20] Werner-Allen, G., Johnson, J., Ruiz, M., Lees, J., Welsh, M.: Monitoring volcanic eruptions with a wireless sensor network. In: Second European Workshop on Wireless Sensor Networks. (2005) 108–120
[21] Werner-Allen, G., Lorinez, K., Ruiz, M., Marcillo, O., Johnson, J., Lees, J., Welsh, M.: Deploying a wireless sensor network on an active volcano
[22] Grossglauser, M., Tse, D.N.C.: Mobility increases the capacity of ad hoc wireless networks. IEEE-ACM Transactions on Networking **10**(4) (2002) 477–486
[23] Wolfson, O., Ouksel, A., Xu, B.: Resource discovery in disconnected mobile ad-hoc networks. In: Proc International Workshop on Next Generation Geospatial Information. (2003)
[24] Wolfson, O., Xu, B., Sistla, A.P.: An economic model for resource exchange in mobile peer to peer networks. In: Proc. 16th International Conference on Scientific and Statistical Database Management (SSDBM'04). (2004)
[25] Wu, Y.H., Guan, L.J., Winter, S.: Peer-to-peer shared ride systems. In Nittel, S., Labrinidis, A., Stefanidis, A., eds.: Advances in Geosensor Networks. Volume 4540 of Lecture Notes in Computer Science. Springer, Berlin (2007)
[26] Estrin, D., Govindan, R., Heidemann, J., Kumar, S.: Next century challenges: scalable coordination in sensor networks (1999)
[27] Rabiner, Heinzelman, W., Kulik, J., Balakrishnan, H.: Adaptive protocols for information dissemination in wireless sensor networks. In: 5th annual ACM/IEEE international conference on Mobile computing and networking, Seattle, Washington, United States, ACM (1999) 174–185
[28] Datta, S., Bhaduri, K., Giannella, C., Kargupta, H., Wolff, R.: Distributed data mining in peer-to-peer networks. IEEE Internet Computing **10**(4) (2006) 18–26
[29] Grossglauser, M., Vetterli, M.: Locating mobile nodes with ease: learning efficient routes from encounter histories alone. Networking, IEEE/ACM Transactions on **14**(3) (2006) 457–469
[30] Estrin, D., Govindan, R., Heidemann, J.: Embedding the internet - introduction. Communications of the ACM **43**(5) (2000) 38–41
[31] Turchin, P.: Quantitative Analysis of Movement: Measuring and Modelling Population Redistribution in Animals and Plants. Sinauer Publishers, Sunderland, MA (1998)