

# Dimensionality Reduction for Long Duration and Complex Spatio-Temporal Queries

Ghazi Al-Naymat  
University of Sydney  
Australia  
ghazi@it.usyd.edu.au

Sanjay Chawla<sup>\*</sup>  
University of Sydney  
Australia  
chawla@it.usyd.edu.au

Joachim Gudmundsson  
National ICT Australia Ltd<sup>†</sup>  
Australia  
joachim.gudmundsson@nicta.com.au

## ABSTRACT

In this paper we present an approach to mine and query spatio-temporal data with the aim of finding interesting patterns and understanding the underlying data generating process. An important class of queries is based on the flock pattern. A flock is a large subset of objects moving along paths close to each other for a certain pre-defined time. One approach to process a “flock query” is to map spatio-temporal data into a high dimensional space and reduce the query into a sequence of standard range queries which can be presented using a spatial indexing structure. However, as is well known, the performance of spatial indexing structures drastically deteriorates in high dimensional space. In this paper we propose a preprocessing strategy which consists of using a random projection to reduce the dimensionality of the transformed space. Our experimental results show, for the first time, the possibility of breaking the curse of dimensionality in a spatio-temporal setting.

## Categories and Subject Descriptors

H.3.3 [Information Search and Retrieval]: Miscellaneous

## Keywords

Spatio-temporal data, data mining, dimensionality reduction

## 1. INTRODUCTION AND RELATED WORK

The most common type of spatio-temporal (ST) data consists of movement traces of point objects. The widespread availability of GPS enabled mobile devices and location-aware sensors have led to an explosion of generation and availability of this type of ST-data.

A unique example of a project which is continuously generating ST-data is related to the tracking of caribou in Northern Canada. Since 1993 the movement of caribous is being tracked through the

<sup>†</sup>National ICT Australia is funded through the Australian Government’s Backing Australia’s Ability initiative, in part through the Australian Research Council.

<sup>\*</sup>The author was partially funded by the Australian Research Council (ARC) Discovery Grant, Project ID: DP055900

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SAC’07 March 11-15, 2007, Seoul, Korea  
Copyright 2007 ACM 1-59593-480-4 /07/0003 ...\$5.00.

use of GPS collars with the underlying expectation that the data collected will help scientist understand the migration patterns of caribous and also help them locate their breeding and calving locations [1]. While the number of caribous tagged at a given time is small the length of the temporal data associated with each caribou is long. As we will see one of the major challenges in ST query processing and patten discovery is to efficiently handle “long duration” ST-data. Some very interesting ST-queries can be formulated on this particular data set. For example, *how are herds formed and does herd membership change over time? Are there specific regions in Northern Canada where caribou herds tend to rendezvous?*

A more sombre example project involving ST-data is related to the study of “pandemic preparedness”. How does a contagious disease get transmitted in a large city given the movement of people across the city? A recent workshop held on spatial data mining<sup>1</sup> exclusively focused on how to use spatial and spatio-temporal data mining techniques to help answer such questions. To abstract the problem we assume that we are given a set  $P$  of  $n$  moving point objects  $p_1, \dots, p_n$  whose locations are known at  $\tau$  consecutive time steps  $t_1, \dots, t_\tau$  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. Since the focus of the paper is to efficiently process known, albeit complex, query patterns, rather than mining for new patterns we will assume that the velocity of an entity along a line segment of the trajectory is constant. The analysis and querying of ST-data is an active research area in many communities, e.g., database [11], GIS and data mining [15], and algorithms [6, 9]. More details can be found in [3].

Laube and Imfeld [13] proposed the REMO framework (RElative MOTion) which defines similar behavior within groups of entities. They define a collection of spatio-temporal patterns based on similar direction of motion or change of direction. Laube et al. [14] extended the framework by not only including direction of motion, but also location itself. They defined several spatio-temporal patterns, including *flock*, *leadership*, *convergence*, and *encounter*, and gave algorithms to compute them efficiently.

However, the above algorithms only consider each time step separately, that is, given  $m \in \mathbb{N}$  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. [6] 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<sub>A</sub> - Given a set of  $n$  trajectories where each trajectory consists of  $\tau$  line segments, a flock in a time

<sup>1</sup><http://www.cs.dartmouth.edu/cbk/sdm06/>

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. Note that  $m, k \in \mathbb{N}$  and  $r > 0$  are given constants.

Assume that the movement of an entity from its position at time  $t_j$  to its position at time  $t_{j+1}$  is described by the straight-line segment between the two coordinates, and that the entity moves along the segment with constant velocity. Benkert et al. [6] proved that there is an alternative, and algorithmically simpler, definition of a flock that is equivalent.

**DEFINITION 2.**  $(m, k, r)$ -flock<sub>B</sub> - Given a set of  $n$  trajectories where each trajectory consists of  $\tau$  line segments a flock in a time interval  $[t_i, t_j]$ , where  $j - i + 1 \geq k$  consists of at least  $m$  entities such that for every discrete time step  $t_\ell$ ,  $i \leq \ell \leq j$ , there is a disk of radius  $r$  that contains all the  $m$  entities.

In the remainder of this paper we refer to Definition 2 whenever we discuss flocks.

## 1.1 Main Contribution and Outline

Current state-of-the-art algorithms for querying complex ST patterns have an exponential dependency on the duration of the ST-data and pattern. In this paper we will show that we can use random projection(s) to manage the exponential dependency while provably retaining a high-quality solution. We will also present experimental results which will confirm that using random projection, as a preprocessing strategy, can effectively help overcome the ‘‘curse of dimensionality’’ for ST pattern processing.

The paper is organized as follows. In Section 2 we briefly go through some interesting ST-patterns for which efficient algorithms would be desirable. The approximation algorithm is presented and analyzed in Section 3. In Section 4 the design of the experiments and the results are discussed. Finally, a summary and directions for future research is given in Section 5.

## 2. SPATIO-TEMPORAL QUERIES

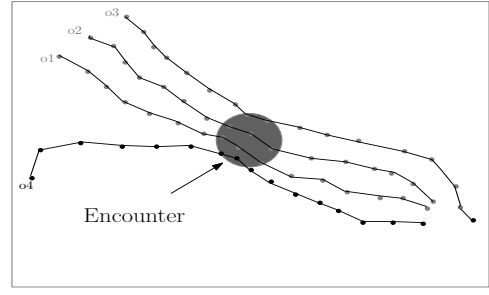
We focus on a class of spatio-temporal patterns proposed by Laube and Imfeld [13]. The class consists of four patterns namely Flock, Leadership, Encounter and Convergence. Other interesting patterns would be Recurrence and Concurrency Recurrence. Our focus will primarily be on the *Flock* query but for completeness we enumerate all of them.

**1: Flock:** Parameters:  $(m > 1, r > 0$  and  $s)$ . At least  $m$  entities are within a circular region of radius  $r$  and they move in the same direction during a time interval of length at least  $s$ . In Figure 1 objects  $o_1, o_2$  and  $o_3$  are participating in a flock pattern.

**2: Leadership:** Parameters:  $(m > 1, r > 0$  and  $\tau > 0)$ . At least  $m$  entities are within a circular region of radius  $r$ , they move in the same direction, and at least one of the entities was already heading in that direction for at least  $\tau$  time steps. In Figure 1, again the three objects  $o_1, o_2$  and  $o_3$  form a leadership pattern with  $o_3$  as the leader.

**3: Encounter:** Parameters:  $(m > 1, r > 0$ , and  $D)$ . At least  $m$  entities will be concurrently inside the same circular region of radius  $r$ , assuming they move with the same speed and direction. In Figure 1, object  $o_1, o_2, o_3$  and  $o_4$  are heading towards an encounter in the shaded region.

**4: Convergence:** Parameters :  $(m > 1, r > 0)$ . At least  $m$  entities pass through the same circular region of radius  $r$  (not necessarily at the same time).



**Figure 1: Spatio-Temporal Patterns**

**5: Recurrence:** Parameters:  $(m > 1, k > 1, r > 1)$ . At least  $m$  entities are visiting a circular region of radius  $r$  at least  $k$  times.

**6: Concurrent Recurrence:** Parameters:  $(m > 1, k > 1, r > 1)$ . At least  $m$  entities are concurrently visiting a circular region of radius  $r$  at least  $k$  times. More figures can be found in [3].

## 3. APPROXIMATING FLOCKS

The nature of the ST patterns is such that they involve ‘‘sufficiently large’’ groups of entities passing a ‘‘small’’ area. This is formalized by the parameters  $m$  and  $r$  in the definitions that represent the number of entities and radius of the region respectively. An exact value of  $m$  and  $r$  has no special significance - 20 caribou meeting in a circle of radius 50 is as interesting as 19 caribou meeting in a circle of radius 51. Therefore the use of approximation algorithms is ideally suited for these scenarios [10].

In this section we generalise the approach suggested by Benkert et al. [6]. The input is a set  $P$  of  $n$  trajectories  $p_1, \dots, p_n$ , where each trajectory  $p_i$  is a sequence of  $\tau$  coordinates in the plane  $(x_1^i, y_1^i), (x_2^i, y_2^i), \dots, (x_\tau^i, y_\tau^i)$ , where  $(x_j^i, y_j^i)$  is the position of entity  $p_i$  at time  $t_j$ .

### 3.1 Previous Approach

The basic idea builds upon the fact that a polygonal line with  $d$  vertices in the plane can be modeled as a point in  $2d$  dimensions. The trajectory of an entity  $p$  in the time interval  $[t_i, t_j]$  is described by the polygonal line  $p(i, j) = \langle (x_i, y_i), \dots, (x_j, y_j) \rangle$ , which corresponds to a point  $p'(i, j) = \langle (x_i, y_i, \dots, x_j, y_j) \rangle$  in  $2(j - i + 1)$ -dimensional space.

The first step when checking whether there is a flock in the time interval  $[t_i, t_{i+k-1}]$  is to map the polygonal lines of all entities to  $\mathbb{R}^{2k}$ .

The second is to characterize the image of the flock in higher dimensional space. It turns out that a flock can be expressed in terms of a high-dimensional *pipe* which we now define.

**DEFINITION 3.** An  $(x, y, i, r)$ -pipe in  $\mathbb{R}^{2k}$  is the following region:  $\{(x_1, \dots, x_{2k}) \in \mathbb{R}^{2k} \mid (x_i - x)^2 + (x_{i+1} - y)^2 \leq r^2\}$ .

An  $(x, y, i, r)$ -pipe is an unbounded region in  $\mathbb{R}^{2k}$  and contains all the points that are only restricted in two of the  $2k$  dimensions (namely in dimensions  $i$  and  $i + 1$ ) and when projected on those two dimensions lie in a circle of radius  $r$  around the point  $(x, y)$ . Equivalence 1 gives the key characterization of flocks.

**EQUIVALENCE 1.** Let  $F = \{p_1, \dots, p_m\}$  be a set of entities and let  $I = [t_1, t_k]$  be a time interval. Let  $\{p'_1, \dots, p'_m\}$  be the mappings of  $F$  to  $\mathbb{R}^{2k}$  w.r.t.  $I$ . It holds that:

$$F \text{ is a } (m, k, r)\text{-flock} \iff$$

$$\exists x_1, y_1, \dots, x_k, y_k : \forall p \in F : p' \in \bigcap_{i=1}^k (x_i, y_i, 2i-1, r)\text{-pipe}.$$

Equivalence 1 has been used by Benkert et al. [6] to design a  $\Delta$ -approximation algorithm to find flocks ( $\Delta > 1$ ) by performing a set of  $n$  range counting queries in the transformed space.

DEFINITION 4. A  $\Delta$ -approximation algorithm will report every  $(m, k, r)$ -flock, may or may not report an  $(m, k, \Delta r)$ -flock and will not report a  $(m, k, r')$ -flock where  $r'$  exceeds  $\Delta r$ .

For each time interval  $I = [t_i, t_{i+k-1}]$ , where  $1 \leq i \leq \tau - k + 1$ , do the following computations. For each entity  $p$  let  $p'$  denote the mapping of  $p$  to  $\mathbb{R}^{2k}$  with respect to  $I$ . For each point  $p' \in P$  perform a range counting query where the query range  $Q(p')$  is the intersection of the  $k$  pipes  $(x_i, y_i, 2i-1, 2r)$  and  $(x_i, y_i)$  is the position of entity  $p$  at time step  $t_i$ . Every counting query containing at least  $m$  entities corresponds to an  $(m, k, 2r)$ -flock according to [6]. Note that the same flock may be reported several times. Since the theoretical bounds for answering range counting queries in high dimensions are close to linear Benkert et al. instead used  $(1+\delta)$ -approximate range counting queries, to obtain the following result.

FACT 1. The algorithm is a  $(2 + \delta)$ -approximation algorithm and requires  $O(\tau n k^2 (\log n + 1/\delta^{2k-1}))$  time and  $O(\tau n)$  space, for any  $\delta > 0$ .

In [6] the skip-quadtrees by Eppstein et al. [8] was used to achieve the theoretical bounds. However, in practice it turns out that a standard quadtree performs slightly better. Let  $S = p_1, p_2, \dots, p_n$  be a set of  $n$  points in the plane contained in a square  $C$  of length  $l$ . A quadtree  $QT$  for  $S$  is recursively constructed as follows: The root of  $QT$  corresponds to the square  $C$ . The root has four children corresponding to the four squares of  $C$  of length  $\frac{l}{2}$ . The leaves of  $QT$  are the nodes whose corresponding square contains exactly one point. Using a compressed quadtree [4] for  $QT$  reduces its size to  $O(n)$  by removing nodes not containing any points of  $S$  and eliminating nodes having only one child. A compressed quadtree for a set of  $n$  points in the plane can be constructed in  $O(n \log n)$  time.

In [6] it was shown that this approach is very effective for small values of  $k$ , they performed experiments with  $k$  in the range of 4 to 16. However, since the running time has an exponential dependency on  $k$  it is obvious that it cannot handle long duration flocks very well.

### 3.2 Using Random Projection

Several techniques are available for carrying out the dimensionality reduction, e.g., PCA (Principle Component Analysis), discrete wavelet transform, discrete cosine transform, and random projection [5]. We have chosen to use random projection technique because: (1) It provides us with pointwise bounds on the distortion (i.e. distance preserving), (2) data can be projected into low dimensions while still retaining high accuracy, and (3) it can be computed efficiently in  $O(\log n)$  time [7].

To generalise the approach that has been mentioned in Section 3.1 to handle long duration flocks, we extend the algorithm by adding a preprocessing step. The added step is intended to reduce the number of dimensions so that we can apply the algorithm in [6]. However, a flock is defined as a group of entities that at all times lie within a disk in the plane of radius  $r$ . If two entities lie within the same query range that means that the distance between them is at most  $2r$  in every dimension. However, the distance between them in  $2k$ -dimensions may be roughly  $r \cdot \sqrt{2k}$ . This observation makes

it clear that we cannot perform a dimensional reduction and then use the same approach as above since the error will be too large.

However, if we modify the definition slightly the generalization can still be performed.

DEFINITION 5.  $(m, k, r)$ -flock<sub>C</sub> - Given a set of  $n$  trajectories where each trajectory consists of  $\tau$  line segments a flock  $f$  in a time interval  $[t_i, t_j]$ , where  $j - i + 1 \geq k$  consists of at least  $m$  entities such that for every pair of entities  $p, q \in f$  it holds that  $\sum_{\ell=i}^j |p_\ell - q_\ell| \leq r \cdot \sqrt{2k}$ .

Instead of using a maximum of  $r$  in each time step we bound the sum of the differences. Intuitively this means that two entities that are very close to each other in all but one time step may still belong to the same flock in the new definition while this would not be possible in the definition by Benkert et al. .

Recall that in a step we consider a set of  $n$  points in  $\mathbb{R}^{2k}$ . Our first approach reduces the dimensions by using random projections by Johnson and Lindenstrauss [12], see also Achlioptas [2]. The following theorem summarizes the tool:

THEOREM 1. [2] Let  $P$  be an arbitrary set of  $n$  points in  $\mathbb{R}^d$ , represented as an  $n \times d$  matrix  $A$ . Given  $\beta, \epsilon > 0$  let

$$k_0 = \frac{4 + 2\beta}{\epsilon^2/2 - \epsilon^3/3} \log n$$

For integer  $k \geq k_0$ , let  $R$  be a  $d \times k$  random matrix with  $R(i, j) = r_{ij}$ , where  $r_{ij}$  are independent random variables from the following probability distributions:

$$r_{ij} = \sqrt{3} \times \begin{cases} +1 & \text{with probability } 1/6 \\ 0 & \dots 2/3 \\ -1 & \dots 1/6. \end{cases}$$

Let

$$E = \frac{1}{\sqrt{k}} AR.$$

Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}^k$  map the  $i^{\text{th}}$  row of  $A$  to the  $i^{\text{th}}$  row of  $E$ . With probability at least  $1 - n^{-\beta}$ ,  $\forall u, v \in P$

$$(1 - \epsilon) \|u - v\|^2 \leq \|f(u) - f(v)\|^2 \leq (1 + \epsilon) \|u - v\|^2$$

Recall that all the trajectories with  $k$  vertices in the plane can be modeled as a point in  $2k$  dimensions. Thus after the transformation we have a set  $P$  of  $n$  points in  $2k$  dimensions. Next, instead of performing  $n$  range counting queries we first apply the random projection on  $P$  to obtain a set  $P'$  of  $n$  points in  $k' = \frac{4+2\beta}{\epsilon^2/2 - \epsilon^3/3} \log n$  dimensions. Then, for each entity  $p$  perform a  $(1 + \delta)$ -approximate range counting query where the query range  $Q(p')$  is the  $k'$ -dimensional ball of radius  $O(\sqrt{k})$  and centre at  $p'$  which is the mapping of  $p$  in  $P'$ .

### 3.3 A Theoretical Analysis

Here we briefly give a probabilistic analysis on how the dimensional reduction affects the algorithm. We will say that an algorithm is an  $\alpha$ -probabilistic  $(m, k, \Delta r)$ -approximation algorithm if every  $(m, k, r)$ -flock is reported with probability  $\alpha$ , an  $(m, k, \Delta r)$ -flock may or may not be reported, while no  $(m, k, \Delta' r)$ -flock will be reported with probability  $\alpha$ .

LEMMA 1. The modified algorithm is a  $(1 - n^{1-\beta})$ -probabilistic  $(m, k, \Delta r)$ -approximation algorithm with running time  $O(\tau n k^2 (\log n + 1/\delta^{2k'}))$  time, where  $k' = \frac{4+2\beta}{\epsilon^2/3 - \epsilon^3/3} \log n$  and  $\Delta = 2(1 + \delta) \cdot (1 + \epsilon)^2$  for any constants  $\epsilon, \delta > 0$  and  $\beta > 1$ .

The proof was omitted because of the space limitation, however it can be found in [3].

## 4. RESULTS AND DISCUSSION

We now report on the experiments that we have carried out to determine how the use of random projections will effect the accuracy of detecting flocks in long duration spatio-temporal data sets. We also report on the overheads caused because of preprocessing the data with a random projection.

### 4.1 Data Set and Parameters

We used a synthetic data set as it allowed us to control the number of flocks present in the data. This helped in determining the correctness of our approach. Twenty data sets with varying number of points, number of flocks and duration of flocks were created. In particular, five data sets each of size 16K, 20K, 32K, 64K and 100K were seeded with 32, 40, 64, 128 and 200 flocks respectively of duration (time step) 8, 16, 500 and 1000. The size of each flock was set to 50 entities and the radius was fixed to 50 units. In the original data (before the random projection), each point coordinate was selected from the interval  $[0, \dots, 2^{16}]$ . In our experiments we always looked for flocks of at least 50 entities in a circle with radius 50 and full time duration (8, 16, 500 or 1000).

### 4.2 Random Projection Parameters

The theoretical lower bound for random projection is of the order  $O(\frac{\ln(n)}{\epsilon^2})$ . Notice that the bound is independent of the dimensionality of the original space. For a data set of size 32K and a distortion of  $\epsilon = 0.5$ , the dimensionality of the projected space will be at least 165. This is too large for the quadtree indexing structure to handle. However, several research studies [7] have noted that the theoretical bound is quite conservative. For example, Bingham and Manilla [7] have noted that for *their* image data the theoretical bound required was  $k \approx 1600$  but a “ $k \approx 50$  was enough.”

In order to test the difference between the theoretically derived and experimentally acceptable bound ( $k$ ) we carried out several experiments for different values of the size of the data set ( $n$ ), dimensionality ( $d$ ), the error tolerance ( $\epsilon$ ), and the confidence ( $\beta$ ). We report on two sets of parameters as shown in Figure 2. Notice that while the theoretical bound depends upon  $n$  (as expected) the experimentally derived bound quickly stabilizes for different values of  $n$ . This confirms that the theoretical bound is too conservative and that for practical applications we can use a much lower  $k$ .

In the end we settled for projecting the data to a 32 dimension space. The primary reason being that the experiments reported in [6] only go up to sixteen time steps (32 dimension) after which the authors note that a range query search using quadtree effectively reduces to a linear scan. Since random projection is being used as a pre-processing step, it is reasonable to project to a space where the use of quadtree is still effective. We may also use a different technique to reduce the number of dimensions. This step is performed on the coordinates obtained from the random projection. Given a positive constant  $\gamma < 1$  randomly choose  $k'' = \gamma k'$  coordinate pairs, thus the point set  $P'$  in  $\mathbb{R}^{2k'}$  can be reduced to a set  $P''$  of  $n$  points in  $\mathbb{R}^{2k''}$ . However, this option is not pursued here.

### 4.3 Results

The results of the experiments are shown in Table 1<sup>2</sup>. The trends are more easily noticeable in Figure 3.

<sup>2</sup>The table has been squeezed because of the space limitations. More details can be found in [3]

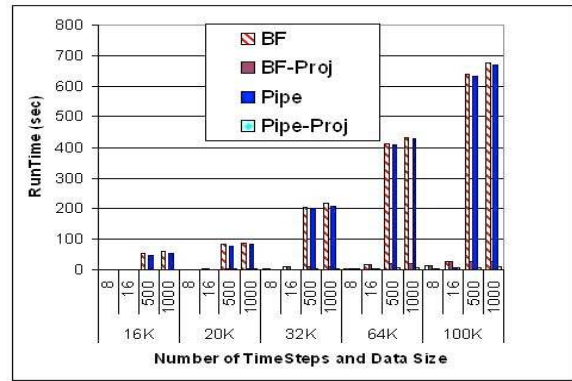


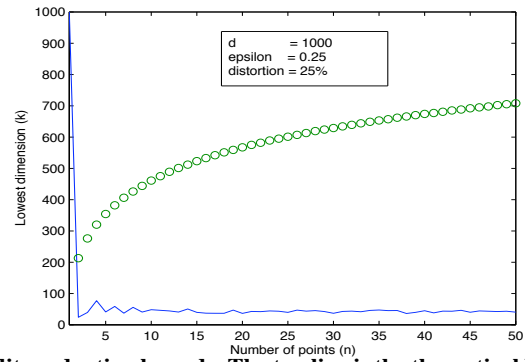
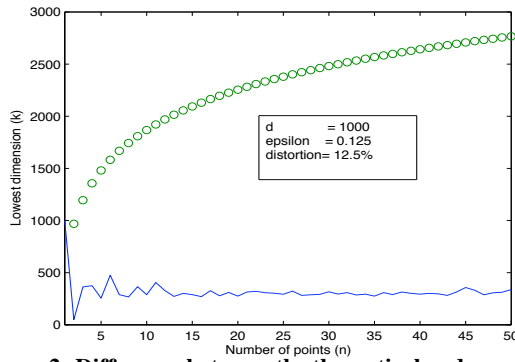
Figure 3: Results- With and Without Random Projection using 16K, 20K, 32K, 64K and 100K data sizes.

The two basic methods we used are *Brute-Force (BF)* and *Pipe*. The BF method does not use any index. It consists of two nested loops, the outer one specifying a potential flock center and inner one computing the distance between a point and the potential flock center. If there are at least  $m$  points within a ball of radius  $2r$  centered at the potential flock center then a flock is reported. In that respect, the BF method is a 2-approximation. Notice that the complexity of the BF method is quadratic in the number of entities and does not depend upon the number of time stamps. This explains the small increase in time (Table 1 in [3], rows 3 and 4) for 16K points from 50 seconds to 59 seconds as the number of time steps (TS) increases from 500 to 1000. The Pipe method is based on Equivalence 1 (Section 3.1) and uses a compressed quadtree as the underlying indexing structure. As expected, the Pipe method beats BF for low dimensions. For example, for data of size 32K, BF takes 2 and 8 seconds to find the 64 flocks for time duration 8 and 16 respectively (rows 1 and 2). For high dimensions (500 and 1000) the quad tree provides no extra advantage because of the “curse of high-dimensionality”: the internal nodes of a quadtree has  $2^d$  children where  $d$  is the number of dimensions. Using 16 timesteps means 32 dimensions which translates to more than 4 billion quadrants. It is very unlikely that the 32K randomly distributed points (not in flocks) fall into the same quadrant. This results in a very flat tree and high running time. We did not apply the random projection pre-processing step for time steps 8 and 16. The overall cost of random projection is  $O(ndk)$  which is around 1 to 3 seconds for the data set 32K and 100K respectively for the largest time steps.

The **most important** result with the random projection is that we retrieve exactly the same number of flocks as retrieved without the random projection. This shows that the distortion induced by the random projection is within acceptable bounds and does not violate the overall correctness.

## 5. SUMMARY AND FUTURE WORK

Given a large spatio-temporal data set, an important query is to retrieve objects which move along paths which are close to each other for a long duration of time, this is called a *Flock*. The previously best known approximation algorithm for such queries have an exponential dependency on the the duration parameter of the query pattern. We have proposed the use of random projection as a practical solution to manage the exponential dependency. We have proved that the random projection will return the “correct” answer with high probability and follow it up with an experimental confirmation on several synthetic data sets. For future work we would



**Figure 2: Difference between the theoretical and experimental dimensionality reduction bounds. The top line is the theoretical bound and the bottom line is derived experimentally using a brute force procedure.**

**Table 1: Results- With and Without Random Projection using 32K, 64K and 100K data sizes.**

Data		Brute Force(BF)		Brute Force-With Proj		Pipe		Pipe-With Proj		
	Data Size	# TS	# Flocks	Sec	# Flocks	Sec.	# Flocks	Sec.	# Flocks	Sec.
1	32K	500	64	206	64	8	64	(BF) 206+	64	3
2	32K	1000	64	216	64	9	64	(BF) 216+	64	3
3	64K	500	128	412	128	18	128	(BF) 412+	128	5
4	64K	1000	128	432	128	19	128	(BF) 432+	128	6
5	100K	500	200	640	200	27	200	(BF) 640+	200	7
6	100K	1000	200	675	200	28	200	(BF) 675+	200	8

like to explore the use of association rule mining type algorithms to discover flock-like patterns. For example, we know that if  $F$  is in  $(m, k, r)$ -flock then every subset of  $F$  of size  $m' < m$  is a  $(m', k, r)$ -flock.

## 6. REFERENCES

- [1] Porcupine caribou herd satellite collar project. <http://www.taiga.net/satellite/>.
- [2] D. Achlioptas. Database-friendly random projections. *J. Comput. Syst. Sci.*, 66(4):671–687, 2003.
- [3] G. Al-Naymat, S. Chawla, and J. Gudmundsson. Dimensionality reduction for long duration and complex spatio-temporal queries. TR 600. ISBN 1-86487-874-6, University of Sydney, July 2006.
- [4] S. Arya, D. Mount, N. Netanyahu, and R. Silverman. An optimal algorithm for approximate nearest searching fixed dimensions. *Journal of the ACM*, 45(6):891–923, 1998.
- [5] A. Bagnall, Chotirat, E. Keogh, and G. Janacek. A bit level representation for time series data mining with shape based similarity. *Data Mining and Knowledge Discovery*, 13:41–65, 2006.
- [6] M. Benkert, J. Gudmundsson, F. Hübner, and T. Wolle. Reporting flock patterns. In *Proc. of the 14th European Symposium on Algorithms*. Springer-verlag, 2006.
- [7] E. Bingham and H. Mannila. Random projection in dimensionality reduction: applications to image and text data. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250, New York, NY, USA, 2001. ACM Press.
- [8] M. T. G. D. Eppstein and J. Z. Sun. The skip quadtree: a simple dynamic data structure for multidimensional data. In *21st ACM Symp. Comp. Geom.*, pages 296–305, Pisa, 2005.
- [9] J. Gudmundsson and M. van Kreveld. Computing longest duration flocks in spatio-temporal data. In *Proc. 14th ACM Symposium on Advances in GIS*, 2006.
- [10] J. Gudmundsson, M. van Kreveld, and B. Speckmann. Efficient detection of motion patterns in spatio-temporal data sets. *To appear in GeoInformatica*, 2006.
- [11] R. Güting, M. Bohlen, M. Erwig, C. Jensen, N. Lorentzos, E. Nardelli, M. Schneider, and J. R. Viqueira. Spatio-temporal models and languages: An approach based on data types. In *Spatio-Temporal Databases: The CHOROCHRONOS Approach*, pages 117–176. Springer-Verlag, 2003.
- [12] W. B. Johnson and J. Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. In *Conference in modern analysis and probability*, pages 189–206. Amer. Math. Soc, 1982.
- [13] P. Laube and S. Imfeld. Analyzing relative motion within groups of trackable moving point objects. In *GIScience '02: Proceedings of the Second International Conference on Geographic Information Science*, pages 132–144, London, UK, 2002. Springer-Verlag.
- [14] P. Laube, M. van Kreveld, and S. Imfeld. Finding REMO – detecting relative motion patterns in geospatial lifelines. In *11th International Symposium on Spatial Data Handling*, pages 201–214, 2004.
- [15] F. Verhein and S. Chawla. Mining spatio-temporal association rules, sources, sinks, stationary regions and thoroughfares in object mobility databases. In *Database Systems for Advanced Applications: 11th International Conference, DASFAA*, pages 187 – 201, Singapore, 2006. Springer Berlin-Heidelberg.