I. Methods

This section explains two alternative methodologies to find moving flock patterns in large spatio-temporal datasets using modern distributed frameworks to divide and parallelize the work load.

A. *The BFE algorithm*

The first alternative follows closely the steps explained at [1]. At this work, the authors proposed the Basic Flock Evaluation (BFE) algorithm to find flock pattern on trajectories databases. The details of the algorithm can be accessed at the source but figure 1 explains schematically the work flow where we can identify 4 general steps:

1) Pair finding: Using the ε parameter, the algorithm query the set of points to get the set of pairs which laid at a maximum distance of ε units. Usually, it is a distance self-join operation over the set of points using ε as the distance parameter. The query also pays attention to do not return pair duplicates. Pair between point p_1 and p_2 is the same that pair between p_2 and p_1 and just one of them should be reported (the id of each point is used to filter duplicates).
2) Center computation: From the set of pairs of points available, each tuple is the input of a simple computation to locate the centers of the two circles of radius $\frac{\varepsilon}{2}$ which circumference laid on the input points. The pseudocode of the procedure can be seen in appendix A.
3) Disk finding: Once the centers have been identified, a query to collect the points around those centers is needed in order to group the set of points which laid ε distance unit each other. This is done by running a distance join query between the set of points and the set of centers using $\frac{\varepsilon}{2}$ as the distance parameter. Therefore, a disk will be defined by its center and the IDs of the points around it. At this stage, a filter is applied to remove those disks which collect less than μ elements around it.
4) Disk pruning: It is possible than a disk collects the same set of points, or a subset, of the set of points of another disk. In such cases, the algorithm should report just that one which contains the others. An explanation of the procedure can be seen in appendix B.

It is important to note that BFE also proposes a grid index structure to speed up spatial operations. The algorithm divides the space area in a grid of ε side (see figure 2). In this way, BFE just processes each grid and its 8 neighborhood grids. It does not need to query grids outside of its neighborhood given that points in other grids are far away to affect the results.

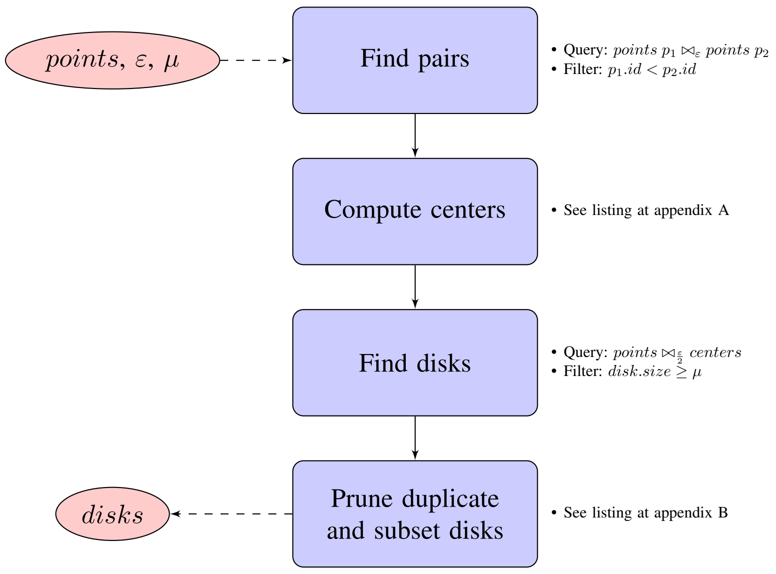
points, ε, μ
Find pairs
• Query: points p₁ ⋈ε points p₂
• Filter: p₁.id < p₂.id
Compute centers
• See listing at appendix A
Find disks
• Query: points ⋈ε/2 centers
• Filter: disk.size ≥ μ
Prune duplicate and subset disks
• See listing at appendix B
disks

Fig. 1. General steps in BFE algorithm.

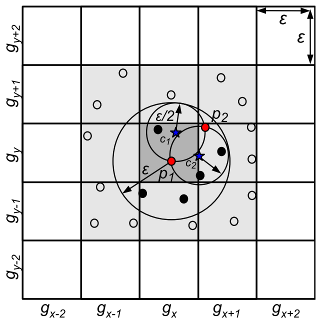

Fig. 2. The grid-based index structure proposed at [1].

Appendix A
Center computation.

Algorithm 1 Find the centers of given radius which circumference laid on the two input points.

Input: Radius $\frac{\varepsilon}{2}$ and points p_1 and p_2.

Output: Centers c_1 and c_2.

```
1:  function FindCenters($p_1$, $p_2$, $\frac{\varepsilon}{2}$)
2:      $r^2 \leftarrow (\frac{\varepsilon}{2})^2$
3:      $X \leftarrow p_1.x - p_2.x$
4:      $Y \leftarrow p_1.y - p_2.y$
5:      $d^2 \leftarrow X^2 + Y^2$
6:      $R \leftarrow \sqrt{|4 \times \frac{r^2}{d^2} - 1|}$
7:      $c_1.x \leftarrow X + \frac{Y \times R}{2} + p_2.x$
8:      $c_1.y \leftarrow Y - \frac{X \times R}{2} + p_2.y$
9:      $c_2.x \leftarrow X - \frac{Y \times R}{2} + p_2.x$
10:     $c_2.y \leftarrow Y + \frac{X \times R}{2} + p_2.y$
11:     return $c_1$ and $c_2$
12: end function
```

Appendix B
Disk prunning.

References

[1] M. R. Vieira, P. Bakalov, and V. J. Tsotras, "On-line Discovery of Flock Patterns in Spatio-temporal Data," in *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2009, pp. 286–295.

Algorithm 2 Prune disks which are duplicate or subset of others.

Input: Set of disks D.

Output: Set of disks D' without duplicate or subsets.

```
 1: function pruneDisks($D$)
 2:     $E \leftarrow \varnothing$
 3:     For Each disk $d_i$ in $D$ do
 4:         $N \leftarrow d_i \cap D$
 5:         For Each disk $n_j$ in $N$ do
 6:             if $d_i$ contains all the elements of $n_j$ then
 7:                 $E \leftarrow E \cup n_j$
 8:             end if
 9:         end for
10:     end for
11:     $D' \leftarrow D \setminus E$
12:     return $D'$
13: end function
```