

## 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 but, before to explain the details of our contributions, we will explain in a general manner the details of the current state-of-the-art to highlight the challenges and drawbacks at the moment to deal with very large spatio-temporal datasets.

### A. The BFE algorithm

The alternatives we will discuss later follows closely the steps explained at [1]. In this work, the authors proposed the Basic Flock Evaluation (BFE) algorithm to find flock patterns on trajectory databases. The details of the algorithm can be accessed at the source but we will explain the main aspects in a general view. It is important to clarify that BFE runs in two phases: firstly, it finds valid disks in the current time instant; secondly, it combines previous flocks with the recently discovered disks to extend them and report them.

The main inputs of the BFE algorithm are a set of points, a minimum distance  $\varepsilon$  which will define the diameter of the disks where the moving entities should lay, a minimum number of entities  $\mu$  at each disk and a minimum duration  $\delta$  which is the minimum number of time units the entities should be keep together to be considered a flock. Based on these inputs, figure 1 breaks down schematically the work flow of this phase where we can identify 4 general steps. The main goal of this phase is to find a set of valid disks at each time instant to allow further combinations with subsequent set of disks coming in the future.

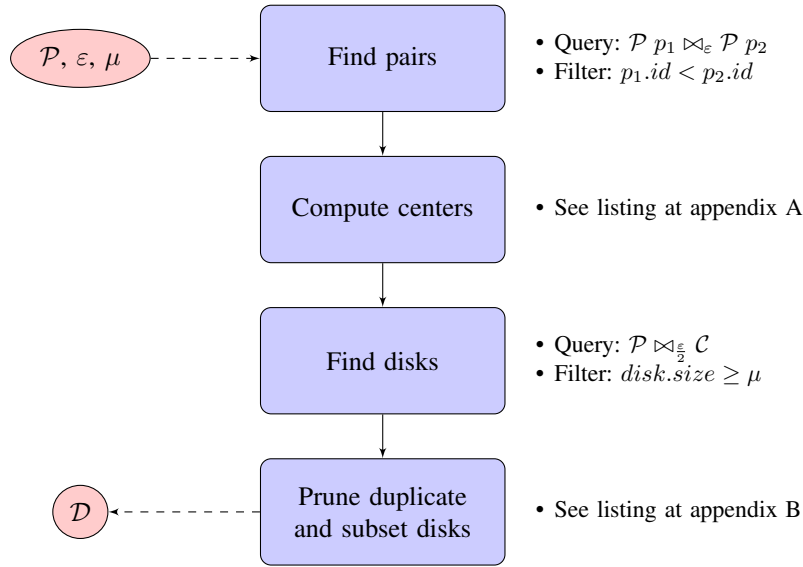


Fig. 1. General steps in phase 1 of the BFE algorithm.

The main steps in phase one can be explained as follows:

- 1) **Pair finding:** Using the  $\varepsilon$  parameter, the algorithm queries the set of points to get the set of pairs which laid at a maximum distance of  $\varepsilon$  units. Usually, it is a distance self-join operation over the set of points using  $\varepsilon$  as the distance parameter. The query also pays attention to do not return pair duplicates. For instance, the pair between point  $p_1$  and  $p_2$  is the same as 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 previous set of pairs, 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  $\varepsilon$  distance units 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  $\mu$  entities around it.
- 4) **Disk pruning:** It is possible that 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 in this phase to speed up spatial operations. The algorithm divides the space area in a grid of  $\varepsilon$  side (see figure 2 from [1]). In this way, BFE just processes each grid and its

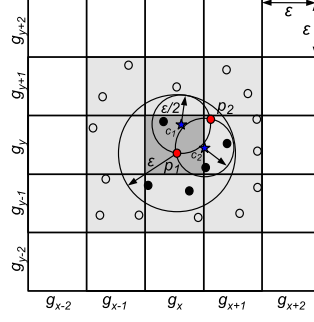


Fig. 2. The grid-based index structure proposed at [1].

8 neighbor 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.

The second phase is more straightforward. Figure 3 explains schematically what is done once the set of current disks (as explained in figure 1) is found at every time instant. This phase performs a recursion using the current set of disks and the previous set of flocks which comes from the previous time instant. Due to we do not know where and how far a group of entities can move in the next time instant, a cross product between both sets is required.

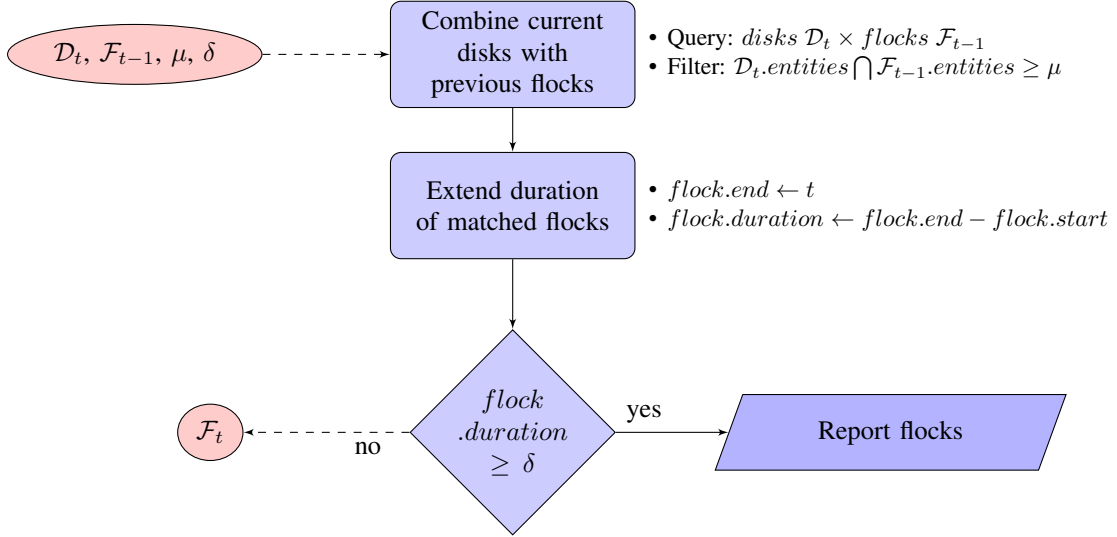


Fig. 3. Steps in BFE phase two. Combination, extension and reporting of flocks.

However, just disks which match the entities of previous flocks are kept. Indeed, only when the size of common items is greater than  $\mu$  we keep the pair. Then, it updates the end time and duration of the filtered flocks. This information is used to decide if it is time to report a flock or keep it for further analysis. If the duration has reached the minimum duration  $\delta$ , a flock is reported and removed from the set. The remaining flocks are sent to the next iteration for further evaluation in the next time instant.

Similarly, figure 4 illustrates the recursion and how the set of flocks from previous time instants feeds the next iteration. The example assumes a  $\delta$  value of 3, so it starts reporting flock since time instant  $t_2$ . Note that time instants  $t_0$  and  $t_1$  are initial conditions. At the very beginning of the execution, we just can find valid disks at  $t_0$  which immediately are transformed to flocks of duration 1 and feed the next time instant. At  $t_1$  we can find a new set of disks  $\mathcal{D}_1$  which combines with the set of previous flocks  $\mathcal{F}_0$ . It updates the information of each flock accordingly but it does not report any flock yet. From now on, subsequent time instants follow strictly the steps summarized on figure 3.

### B. Bottlenecks in BFE and possible solutions

There are some steps during the execution of BFE which are particularly affected when it deals with very large datasets. Firstly, we will focus on phase 1 of BFE. Figure 5, the steps of this phase are illustrated for a sample dataset. You can see that the number of centers and disks found is considerable large in comparison with the final set of valid disks. Indeed, the finding of centers and posterior operations grows exponentially depending on the number of points and possible pairs (which itself depend on the  $\epsilon$  parameter).

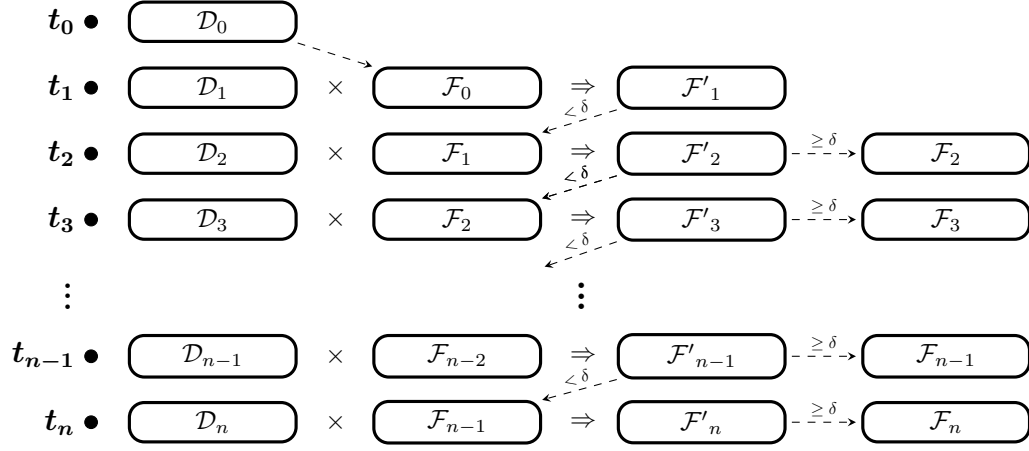


Fig. 4. BFE phase 2 example explaining the recursion of the set of flocks along time instants and the initial conditions.

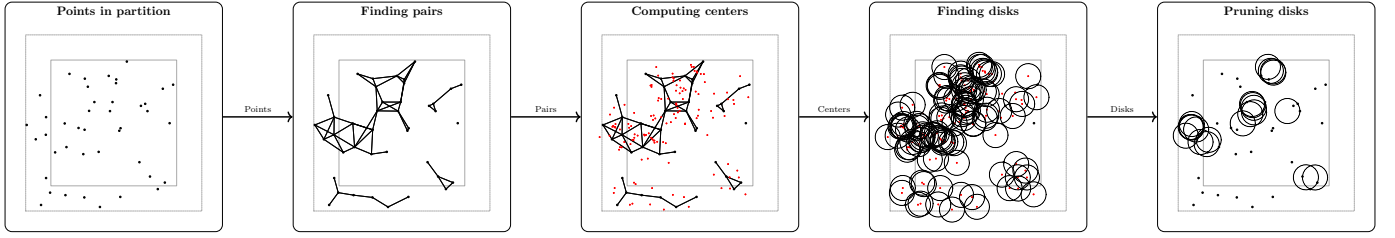


Fig. 5. Example of BFE execution on a sample dataset..

### C. A parallel approach

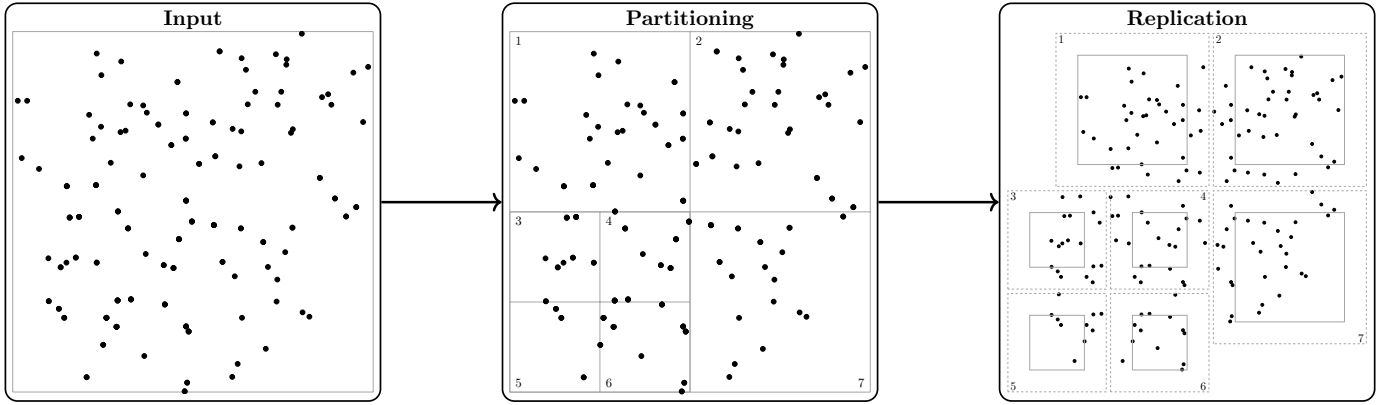


Fig. 6. An example of partitioning and replication on a sample dataset.

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 PRUNING.

---

**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:    $E \leftarrow \emptyset$ 
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

```

---

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.