Flock record

Flock(items, start, end, center):

- items: Set[Int]. Set of point identifiers.
- start: Int. Time instant when the flock starts.
- end: Int. Time instant when the flock finishes.
- center: Point. Geometric representation of the center of the flock.

Pseudo-code

Algorithm 1 Reporting current flocks

Require: Set of previous flocks $\mathcal{F}'$, Set of current maximal disks $\mathcal{D}$, a maximal distance a flock can travel between time intervals θ, minimum distance between points in a flock ε, minimum flock size μ and minimun flock duration δ.

1: $\mathcal{R} \leftarrow$ apply DISTANCEJOIN operation between $\mathcal{F}'$ and $\mathcal{D}$ using θ as distance constraint. ▷ Joining phase
2: $\mathcal{F}' \leftarrow \emptyset$
3: **for each** $r : (disk, intersected\_flocks)$ in $\mathcal{R}$ **do**
4: $new\_flocks \leftarrow \emptyset$
5: $flock1 \leftarrow$ convert $disk$ into a flock ▷ start and end become the current time instant.
6: **for each** $flock2$ in $intersected\_flocks$ **do**
7: $i \leftarrow flock1.items \cap flock2.items$
8: $s \leftarrow flock2.start$
9: $e \leftarrow flock1.end$
10: $c \leftarrow flock1.center$
11: **if** $i.size >= \mu$ **then**
12: $new\_flocks \leftarrow new\_flocks \cup Flock(i, s, e, c)$
13: **end if**
14: **end for**
15: $subsets \leftarrow \emptyset$
16: **for each** f_1 in new_flocks **do**
17: **for each** f_2 in new_flocks **do**
18: **if** $f_1.items.size < f_2.items.size$ **then**
19: **if** $f_1.items$ is subset of $f_2.items$ and $f_1.start >= f_2.start$ **then**
20: $subsets \leftarrow subsets \cup f_1$
21: **end if**
22: **end if**
23: **end for**
24: **end for**
25: $\mathcal{F}' \leftarrow \mathcal{F}' \cup (new\_flocks - subsets)$
26: **end for**
27: $\mathcal{F}_\delta \leftarrow \emptyset$ ▷ Reporting phase
28: **for each** $flock$ in $\mathcal{F}'$ **do**
29: **if** $flock.end - flock.start + 1 = \delta$ **then**
30: $\mathcal{F}_\delta \leftarrow \mathcal{F}_\delta \cup flock$
31: **end if**
32: **end for**
33: $\mathcal{F}_\delta \leftarrow$ REMOVEPOSSIBLEREDUNDANTS$(\mathcal{F}_\delta, \epsilon)$
34: Report flocks in $\mathcal{F}_\delta$

Algorithm 2 REMOVEPOSSIBLEREDUNDANTS function

Require: Set of flocks $\mathcal{F}$ and a minimun distance ε.

```
1: function REMOVEPOSSIBLEREDUNDANTS ( F, ε )
2:     R ← apply SELF-DISTANCEJOIN operation on F using ε as distance constraint.
3:     subsets ← ∅
4:     for each r : (flock, intersected_flocks) in R do
5:         for each flock2 in intersected_flocks do
6:             if flock.size < flock2.size then
7:                 if flock.items is subset of flock2.items then
8:                     subsets ← subsets ∪ flock
9:                 end if
10:            end if
11:        end for
12:    end for
13:    return (F − subsets)
14: end function
```