Supplement

June 2019

1 Theorems

1.1 Triangle cells Vs Square cells for Range query

Let a_{sq} be area of square cell, a_{tr} be area of triangular cell given that, side of cell is ϵ (flock daimeter). Assume that points are distributed equally over the entire area.Hence area under computation is directly proportional to computation complexity. Calculations:

- For range query in square cell, Figure 2 , we need to consider each point for calculating ϵ-neighbourhood in 9 cells (in yellow).Therefore total search space for neighbourhood calculation is,
  $= 9 * \epsilon * \epsilon$
  $= 9 * a_{sq}$

- For triangular cells, there are 2 cases based on location of point p ,

  1. p lies on any vertex of triangular cell Figure 3a,
     All points marked in yellow triangles are at a distance $<= \epsilon$. Therefore all those points can be directly added to range. For the points in the triangles marked blue, we need to check for ϵ-neighbourhood . Thus search space under calculations is reduced to 2 triangles. $= 2 * a_{tr}$
     $= 2 * 0.5 * \epsilon * \sqrt{3}/2 * \epsilon$
     $= \epsilon * \sqrt{3}/2 * \epsilon$
     $=0.866 * a_{sq}$

  2. If p lies on face of triangle or any edge Figure 3b,
     All points marked yellow are directly taken and ones in blue required ϵ-neighbourhood calculation.During this calculation, we always check the x coordinate, if it exceeds p.x+ϵ then search for that triangle is halted. .For any point p on face of triangle, maximum triangles under blue will be 7.5, Therefore search space is,
     $=7.5 * a_{tr}$
     $= 7 * 0.5 * \epsilon * \sqrt{3}/2 * \epsilon$

$= 3.5 * \epsilon * \sqrt{3}/2 * \epsilon$
$= 3.03 * a_{sq}$

Both cases of triangular cells have less search space than square cells for range query.

1.2 Any two points on line segment, and either of the point is to be reached then ,there is higher probability of reaching from midpoint than from start of segment.(Given that two points cannot interchange the order)

Say line segment is pp' having midpoint at m. Let k be mid point of pm. Let a ,b be the point that is to be reached.
There are 4 cases ,

- a, b lie on either sides of m (figure 4a)
  Probablity that a lies left to k given (b-m) > (a-p) = 0.25(a is reachable faster from p)
  Probablity that a lies left to k given (b-m) < (a-p) = 0.25(b is reachable faster from m)
  Probablity that a lies right to k given it is in left of m = 0.5 (a is reachable faster from m)

- a, b lie on right side of m (figure 4b)
  Probablity of m reaching a faster =1
  Probablity of p reaching a faster =0

- a, b lie on left side of m (figure 4c,4d)
  b is never faster reachable from p, hence, there are 2 cases, Probablity that a is faster reachable from p=0.25 Probablity that a is faster reachable from m=0.75

Hence probablity of m reaching either a or b faster
=0.33(0.75+1+0.75)
=0.825
and probablity of p reaching a faster is,
=0.33(0.25+0+0.25)
=0.165
Hence for two points on line segment, and either of the point is to be reached then ,there is higher probability of reaching from midpoint than from start of segment.(Given that two points cannot interchange the order)

2 Triangular grid Construction

Definition 2.1 ***Triangular grid:*** *It is set T_I of triangles t such that for any triangle t in a virtual underlying grid T_V that has at least one point of data set is enclosed t.*

$$T_I = \{ t \mid t \, \epsilon \, T_V \; and \; p \, \epsilon \, P \; and \; t \; = \; encloses(p, t) \; \}$$

The encloses function takes the point p as coordinates x, y as the input and

input : point p(x,y,t), time t;
Result: Triangle T such that p lies inside T
initialize T;
height=$\epsilon * \sqrt{3/2}$;
rownum = floor(p.y/height);
if *rownum is even* **then**
  x1=floor(x/ϵ) * height;
else
  if x_i $\epsilon/2$ **then**
    x1=-$\epsilon/2$
  else
    x1=floor(x/ϵ) * $\epsilon - \epsilon/2$
  end
end
y1=y2=y;
x2=ϵ + x1;
a = angle formed by p(x,y) with (x1,y1);
b = angle formed by p(x,y) with (x2,y2);
T= identify the triangle using a, b as specified in Figure 1b;
return T;

Algorithm 1: Finding enclosing triangle from a given point.

finds the triangle in which it lies. The y-coordinate gives the row number and x coordinate gives exact triangles to which the point belongs. Triangle 1, 2, 3 in figure 1b are the three possibilities of triangles to which the point can belong, when range identified by x is J(x1,y1) and K(x2,y2). Using the angles a, b respectively formed by J, K with a given point p, we can select the final enclosing triangle from the three triangles using following cases.

1. $a \; <= \; 60 \; and \; b \; > \; 120 \; .... \; Triangle \; 2$
2. $a \; <= \; 60 \; and \; b \; <= \; 120 \; .... \; Triangle \; 3$
3. $a \; > \; 60 \; and \; b \; >= \; 120 \; .... \; Triangle \; 1$

3 Range query for any point p

Range query is finding neighbouring points of p such that they lie within range ϵ of p. When triangular grid is used for range query there are two possibilities,

- Range R' of a point p when p lies of vertex of any triangle then,

$$R'(p) = \{p_t \mid p_t \; \epsilon \; t_i, t_o \; and \; distance(p_t, p) <= \epsilon\}$$

  where $t_i,\ t_o$ are internal and outer triangles in figure 2a.

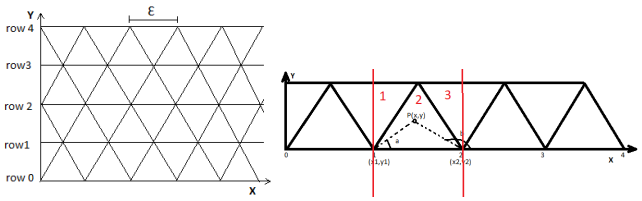
- Similarly if Range R" of a point p which lies withing the triangle or on edges of triangle then,

$$R"(p) = \{p_t \mid p_t \; \epsilon \; t_i, t_o \; and \; distance(p_t, p) \; <= \; t\}$$

  where $t_i,\ t_o$ are internal and outer triangles in figure 2b.

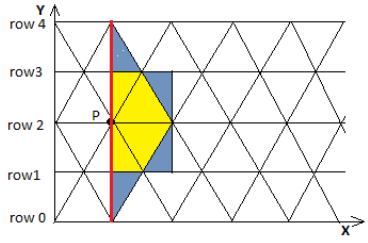
From the range selected for each point, clusters are created with diameter ϵ using approach discussed by M. Vieira et al. [**?**].

4 midstart approach

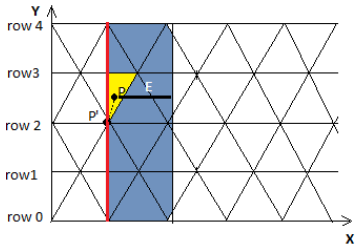


(a) (b)

Figure 1: Triangular Grid Construction, (a) Virtual Triangular grid, (b) Triangle Identification from any point p(x,y)



(a)



(b)

Figure 2: Figure depicting search space of point p for finding the range. (a) shows the case where p lies on vertex of the triangle, (b) shows the case where p lies within face of the triangle or on any edge. Area in yellow denotes the area which is implicitly under the range of p. Area in blue denotes that area which requires explicit calculation of $\epsilon - neighbourhood$. E=ϵ is length of triangle side

```
Input : Clusters C of database snapshots from i to δ + i;
Result: Flock F
initialize start = i;
end = δ + i;
if end-start=0 then
    return F
else
    for c_e in C_end do
        for c_s in C_start do
            F_I = F_I ∪ (c_e ∩ c_s)
        end
    end
    if end-start=1 then
        return F_I;
    else
        F_IN=mid-start(C, i+1);
        if F_IN is empty then
            return F_IN
        end
        for c_n in F_IN do
            for c_i in F_I do
                F= F ;
                ∪ (c_n ∩ c_i)
            end
        end
        return F;
    end
end
```

Algorithm 2: mid-start Algorithm