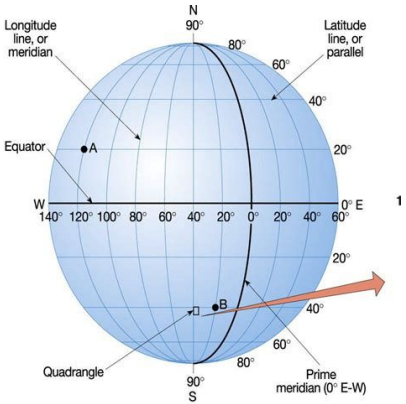
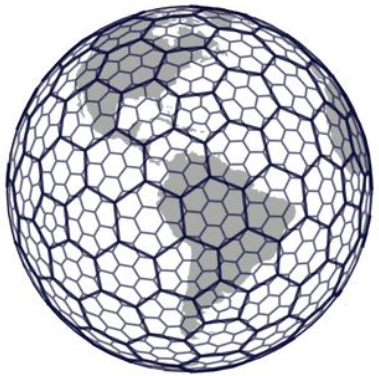
H3-index Solution

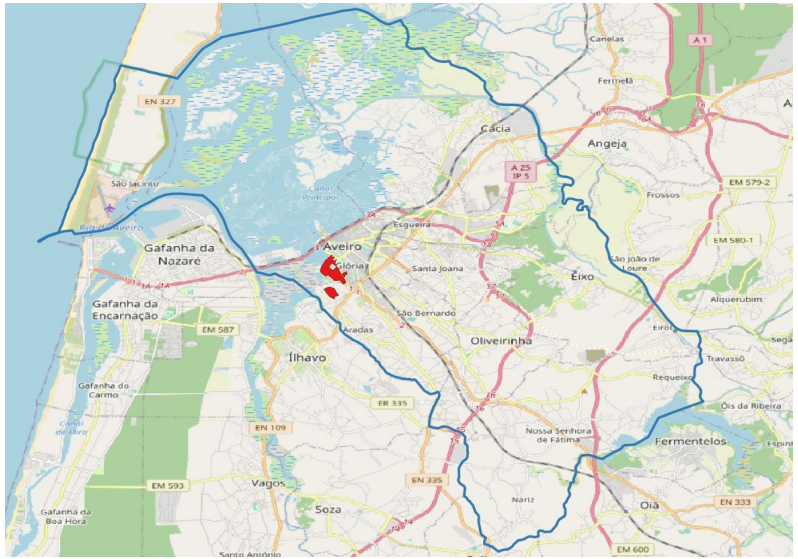
efficient indexation





Problem Definition

- How can we determine if the issue reported by user W at coordinates (X, Y) pertains to organization Z?



Existing Solutions

Ray Casting Algorithm Definition

Input: A point (x, y) and a list of polygons, where each polygon is defined by a sequence of vertices (x_1, y_1), (x_2, y_2), ..., (x_n, y_n).

Process:
For each polygon, check if the point lies inside it:

1. Imagine casting a horizontal ray from the point to the right (or any direction).
2. Count the number of times this ray intersects the edges of the polygon.
3. If the number of intersections is odd, the point is inside the polygon. If even, it's outside.

Edge Case Handling: Points exactly on an edge or vertex.

Can it scale up to **100 organizations**?

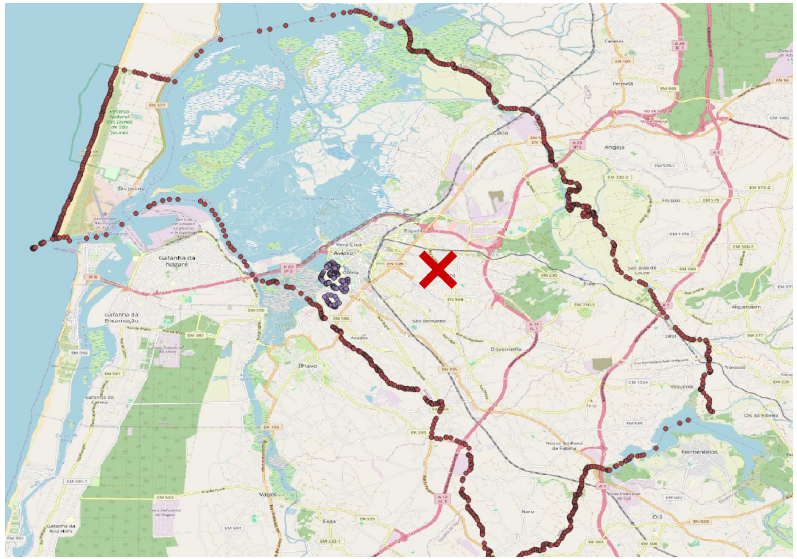
Simple Problem Statement:

- You have 100 polygons (regions). **Not nested**!
- Each polygon has 500 vertices, implying 500 edges
- You want to check which polygon contains a single given point (x, y).
- This requires iterating 100 * 500 = 50,000 times.

Certain optimizations can improve bounding box checks…

Problem

- Region indexation

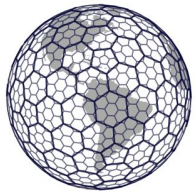


Aveiro: **939 points**
Universidade de Aveiro: **169 points**

(40.630451, -8.6551867) is inside which organization?

H3-Index Solution

Space filling - Contextualization



Solution: What about changing the indexation diagram?

- **Origin**: A geospatial indexing system created by Uber for optimized location-based solutions.
- **Structure**: Divides Earth's surface into a hierarchical grid of hexagons, from global to meter-level precision.
- **Purpose**: Enables efficient storage, retrieval, and analysis of spatial data across varying resolutions.
- **How It Works**: Assigns unique indexes to hexagons, supporting fast queries and scalable compression.
- **Applications**: Drives real-world solutions like ride pricing, fleet optimization, geographic analysis, and our specific use case.

Example: Space filling

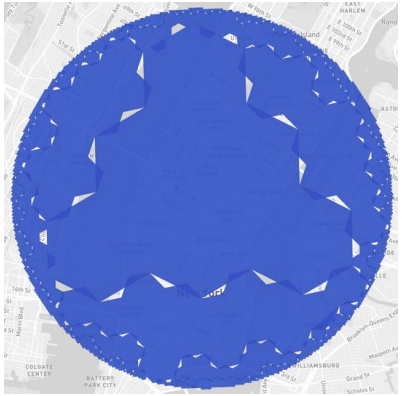
Uber

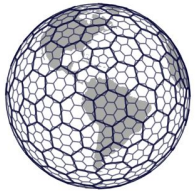
Space filling - Hexagon Advantages

Solution: What about changing the indexation diagram?

- **Uniform Neighbor Distance**: Consistent spacing simplifies spatial analysis.
- **Circle-Like Shape**: Better for modeling nearest-neighbor queries.
- **Efficient Tiling**: Covers Earth with minimal distortion or gaps.
- **Hierarchical Scale**: Flexible resolution for data aggregation or detail.
- **Strong Adjacency**: Six neighbors enhance clustering and flow modeling.
- **Fast Indexing**: O(1) lookup speeds up localization area discovery.

All six neighbors of a hexagon (ring 1)

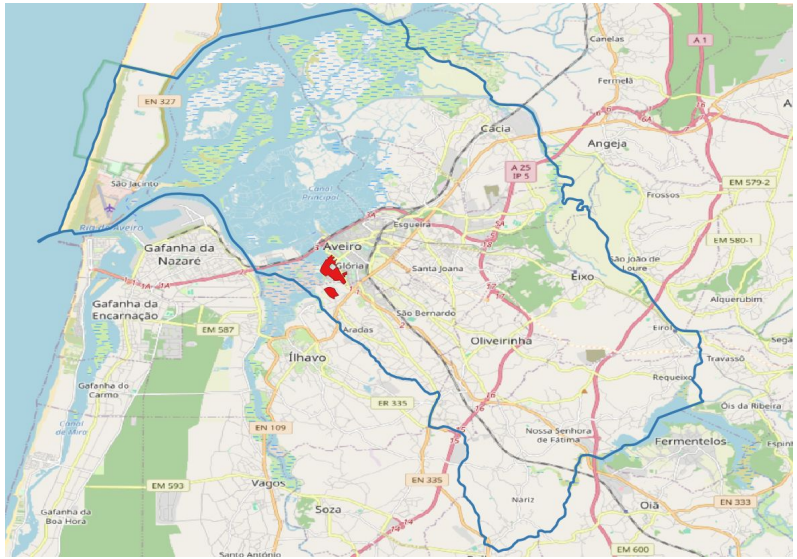




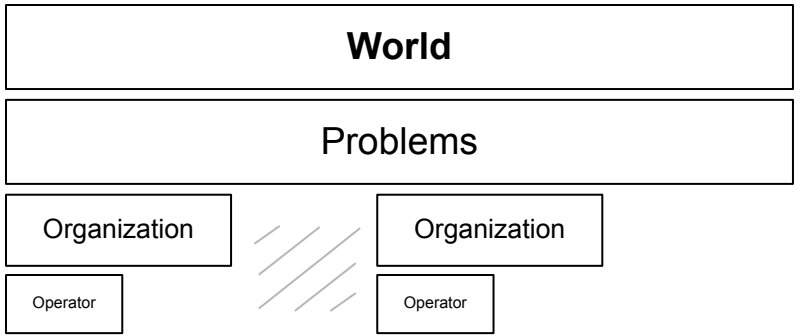
Uber

Our Solution

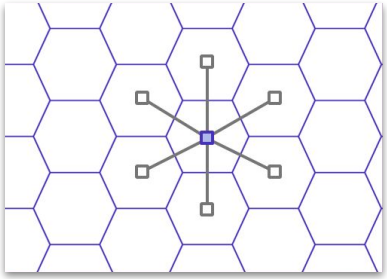
- Hierarchical Information

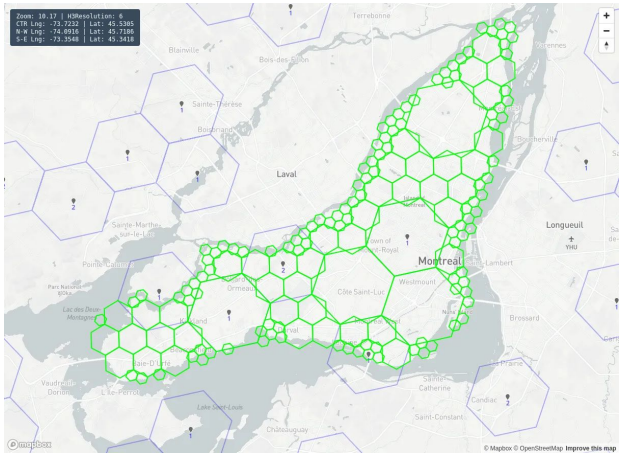


- Can we accept **nested** organizations? **Yes**
- Can we accept **multiple** organizations? **Yes**



H3 Index





(lat, lon) –> H3 cell

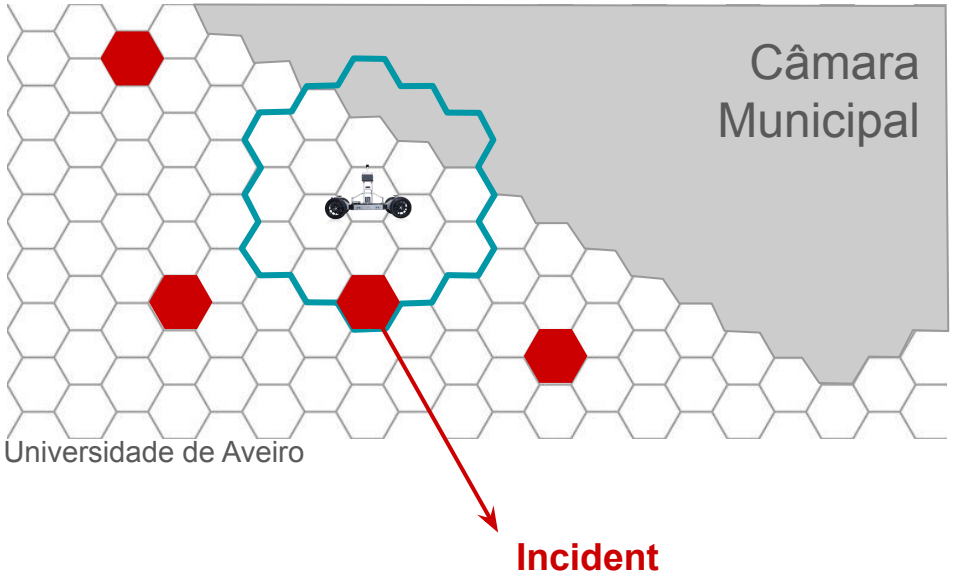
Average area in m^2

Here are the same areas, but in m^2.

Res	Average Hexagon Area (m^2)	Pentagon Area* (m^2)
0	4,357,449,416,078.392	2,562,182,162,955.496
1	609,788,441,794.134	328,434,586,246.469
2	86,801,780,398.997	44,930,898,497.879
3	12,393,434,655.088	6,315,472,267.516
4	1,770,347,654.491	896,582,383.141
5	252,903,858.182	127,785,583.023
6	36,129,062.164	18,238,749.548
7	5,161,293.360	2,604,669.397
8	737,327.598	372,048.038
9	105,332.513	53,147.195
10	15,047.502	7,592.318

Uber

H3 Index Solution



Solution:

key: h-index
value: (organization, incident_id)

Problem Definition

User:
- user_id
- name
- email
- hash_password
- email_notification_flag

Operator:
- operator_id
- email
- hash_password
- organization_id

Organization:
- organization_id
- language
- **region???**

Incident:
- incident_id
- category
- main_description
- first_occurence_date
- centroid_location
- **location???**
- num_ocurrences
- severity
- status

Occurrence:
- occurrence_id
- photo_id
- **photo_location???**
- description
- date
- user_id
- incident_id

Architecture