Cómo hacer un motor de optimización para el mundo real: Python y otras bestias

Python Sevilla

22 de marzo de 2018

www.geographica.gs

Índice

1/

¿Quienes somos?

Geographica

En Geographica **somos unos entusiastas de los datos espaciales, el diseño y el software libre.**

https://geographica.gs

Alberto Asuero
Chief Technology Officer
@alasarr
alberto@geographica.gs

Cayetano Benavent
Head of Data
@cayetanobv
cayetano@geographica.gs

Josema Camacho
Chief Innovation Officer
@josemazo
josema@geographica.gs

2/

Presentación del problema

Optimización de rutas

→ Construir un motor de optimización de rutas es una tarea muy compleja.

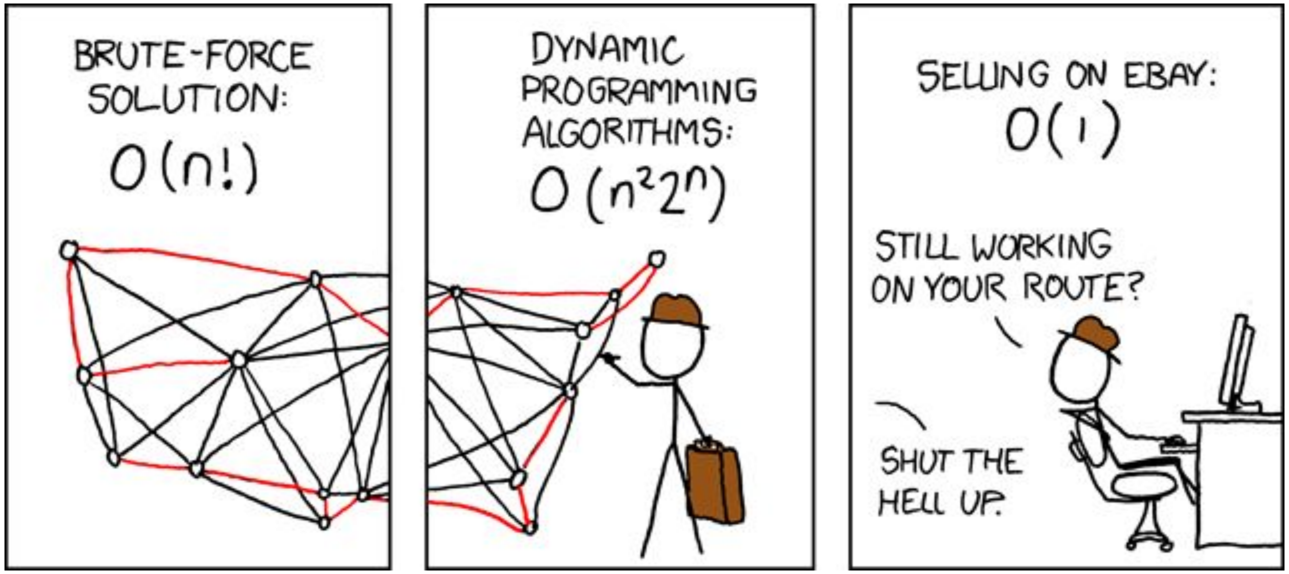
Optimización de rutas

- Este tipo de problemas se conocen como VRP (Vehicle Routing Problems).
- Los problemas VRP se encuadran dentro de la optimización combinatoria, una rama de la optimización matemática.

https://en.wikipedia.org/wiki/Vehicle_routing_problem

Optimización de rutas

→ Los problemas VRP del mundo real no pueden resolverse por fuerza bruta (NP-hard). Hay que buscar otras vías de resolución (más adelante se verá)...



Optimización de rutas

- Hay muchas variantes de problemas VRP (*en el mundo real sueles encontrar variantes y mezclas entre todos ellos*):
  - Capacitated Vehicle Routing Problem (CVRP).
  - Vehicle Routing Problem with Time Windows (VRPTW)
  - Vehicle Routing Problem with Pickup and Delivery (VRPPD).
  - Vehicle Routing Problem with Multiple Trips (VRPMT)
  - Etc.

3/ Nuestra aproximación al problema

Aproximación faseada al problema

- Solemos dividir la resolución del problema en cuatro grandes Fases:

  - Fase 1. Los datos (Fase ETL).
  - Fase 2. La red: topología y costes.
  - Fase 3: La optimización.
  - Fase 4: Construcción de resultados.

¿Por qué Python?

- Python no es la única tecnología usada, pero sí la más importante por ser:
  - Perfecto para prototipar y testar algoritmos.
  - Muy versátil y maduro para construir arquitecturas muy complejas.
  - Es el mejor uniendo tecnologías muy distintas.
  - etc.

Workflows complejos y computación distribuida

- Este tipo de problemas requieren construir workflows complejos.
- Cuando el problema crece mucho, la complejidad computacional se dispara, por lo que soluciones de computación distribuida son necesarias.

Más adelante veremos cómo hemos solventado ambas cuestiones.

4/

Datos y redes...

Los datos

- Esta primera fase involucra todos los procesos relacionados con la extracción, transformación y carga de datos en el sistema (Fase ETL).
- Los conjuntos de datos involucrados son generalmente dos:
  - Cartografía base sobre la que construir la topología de la red.
  - Datos a optimizar. Un ejemplo sería:
    - Conjunto de puntos de entrega/recogida.
    - Posiciones de las cocheras.
    - Posiciones intermedias de descarga o intercambio.
    - etc.

Los datos

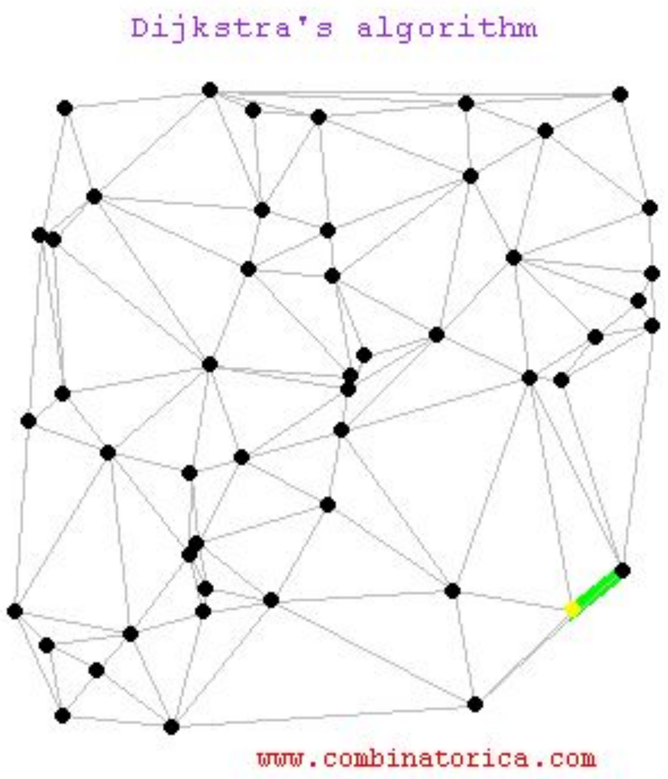
- La calidad de la cartografía base sobre la que construir la red es clave.
- Trabajar con datos abiertos ofrece una flexibilidad enorme: Open Street Map (OSM).
- Los resultados con OSM, en nuestra experiencia, son excelentes.
- Hay infinidad de librerías open source para trabajar con OSM.

La red: topología y costes

- La construcción de la topología de la red parte de los datos de la fase 2.
- La red está compuesta por ejes y nodos, y puede tener dirección (lo habitual con vehículos).
- Un proceso clave es la proyección de las ubicaciones a optimizar como nodos sobre la red.
- El objetivo final de la Fase 2 debe ser proporcionar a la Fase 3 (optimización) el coste que supone recorrer cada eje nuestra red.

La red: topología y costes

- Computar el coste entre dos nodos de la red supone calcular el camino más corto entre ambos nodos (con menor coste, según el criterio que usemos de coste).
- El resultado final de esta fase se traduce en lo que se conoce como matriz de coste.



La red: topología y costes

- Hay mucho software y muy bueno para el cálculo de los costes de la red:
  - https://github.com/pgRouting
  - https://github.com/Project-OSRM
  - https://github.com/valhalla
  - Etc.

- La elección depende obviamente del objetivo de nuestro trabajo, así como de la restricciones que tengamos.

La red: topología y costes

→ El que quiera experimentar con PgRouting, puede usar esta librería Python que genera un entorno dockerizado con todo lo necesario para ello:

https://github.com/GeographicaGS/osm-itinera

5/

Algoritmos de optimización

Objetivo

Optimización = Maximizar y/o minimizar

Máximo dinero y mínimo esfuerzo

Soluciones

- Sistema experto
- Machine Learning
- Fuerza bruta, por ser un problema combinatorio

TSP de 100 ciudades

Soluciones

- Sistema experto
- Machine Learning
- Fuerza bruta, por ser un problema combinatorio

TSP de 100 ciudades

$$100! \approx 9.33 * 10^{157}$$

Soluciones

- Sistema experto
- Machine Learning
- Fuerza bruta, por ser un problema combinatorio

TSP de 100 ciudades

$$100! \approx 9.33 * 10^{157}$$

$$9.33 * 10^{157} >>> 10^{80}$$

Soluciones

- Sistema experto
- Machine Learning
- Fuerza bruta, por ser un problema combinatorio

TSP de 100 ciudades

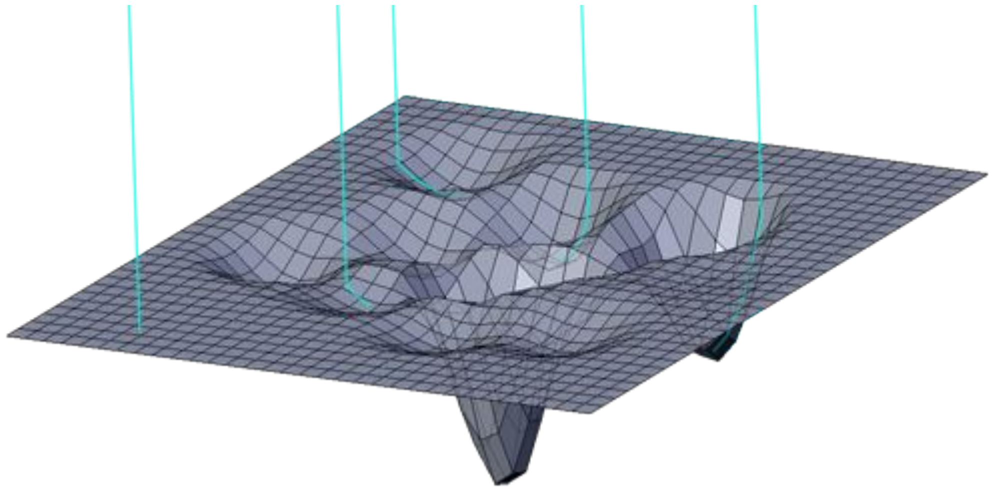
$$100! \approx 9.33 * 10^{157}$$

$$9.33 * 10^{157} >>> 10^{80}$$

10^{80} = número estimado de átomos en el universo visible

Metaheurísticas

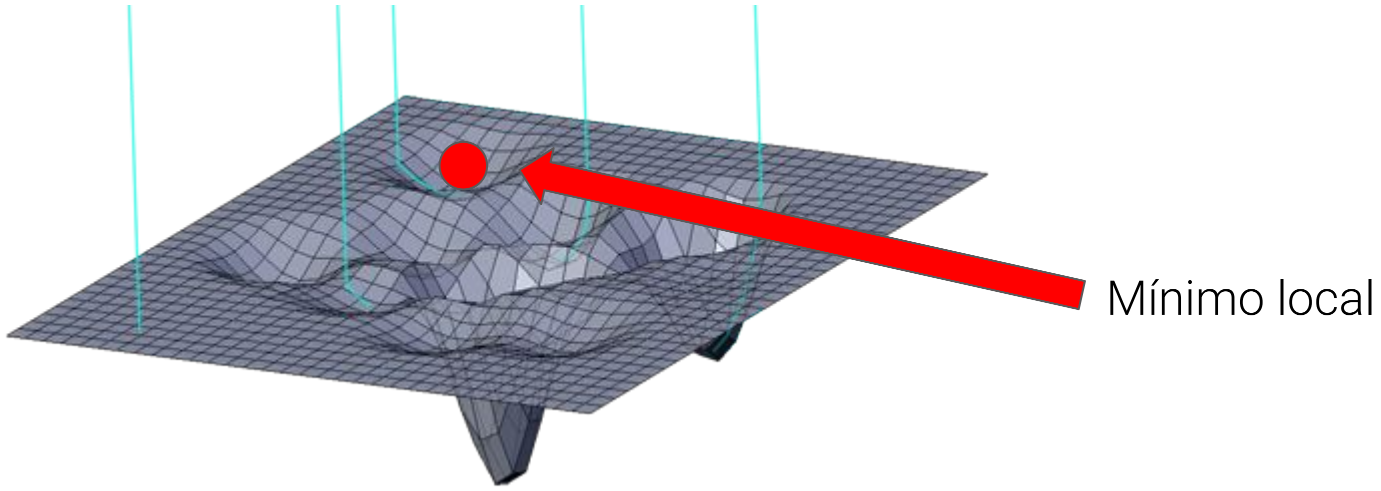
- Navegan por el espacio de soluciones



- Tipos:
  - Población *vs* solución única
  - Artificiales *vs* inspirados en la naturaleza

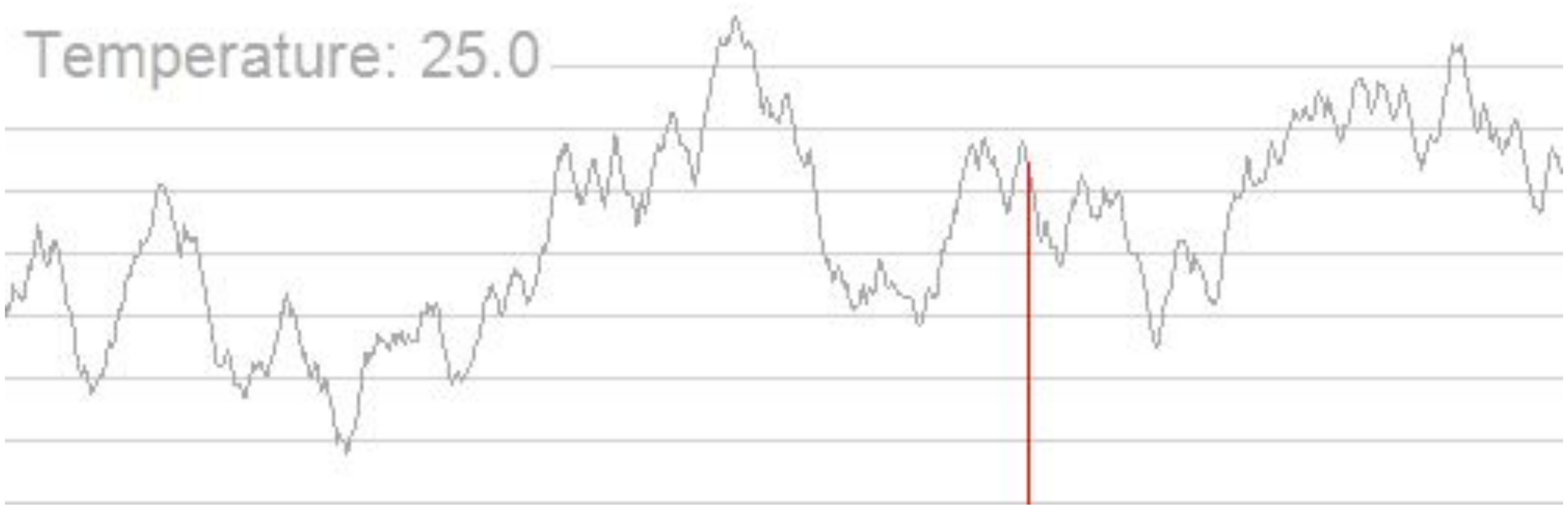
Metaheurísticas

- Navegan por el espacio de soluciones



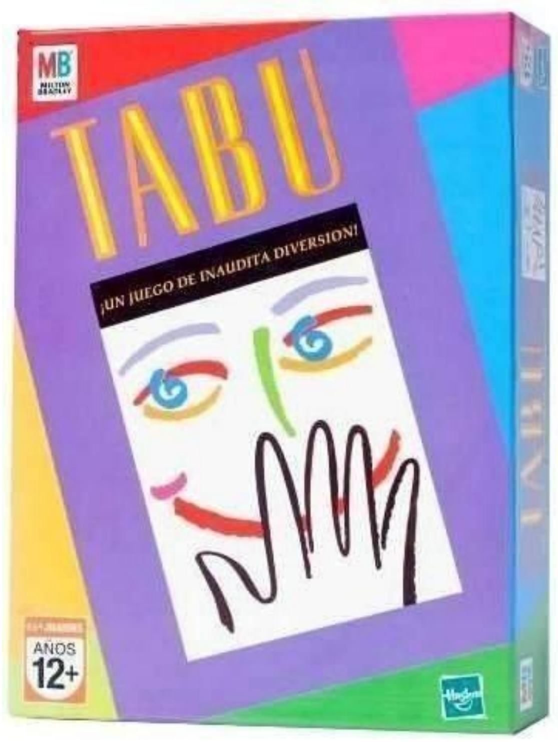
- Tipos:
  - Población *vs* solución única
  - Artificiales *vs* inspirados en la naturaleza

Metaheurísticas



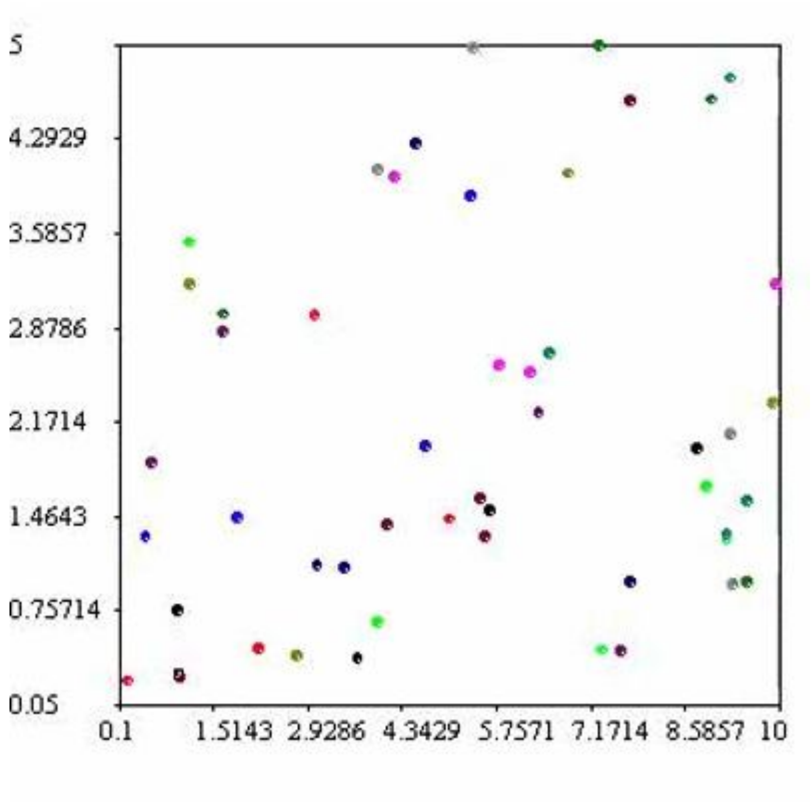
Enfriamiento simulado

Metaheurísticas



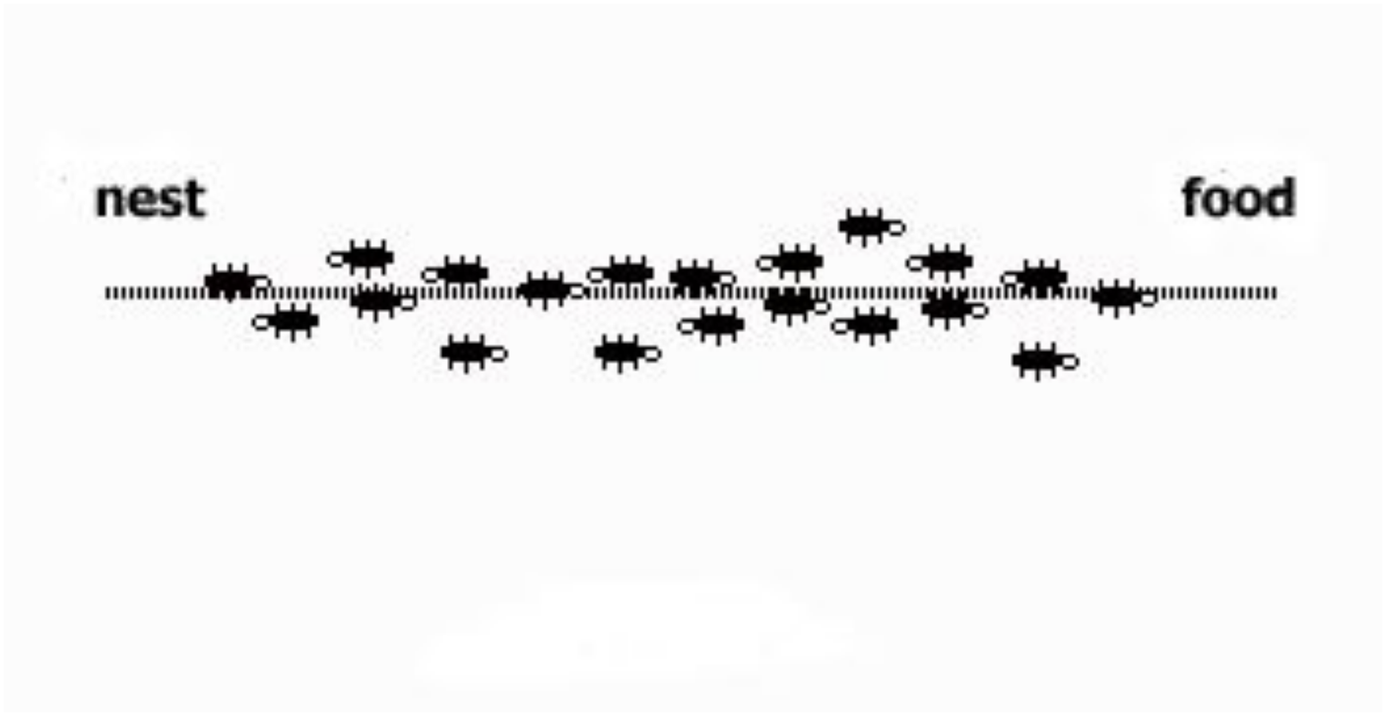
Taboo search

Metaheurísticas



PSO (optimización por enjambre de partículas)

Metaheurísticas



Optimización por colonia de hormigas

A desarrollar metaheurísticas…

VS

Un paseo por GitHub

- → https://github.com/google/**or-tools**
- → https://github.com/graphhopper/**jsprit**
- → https://github.com/kiegroup/**optaplanner**
- → https://github.com/VROOM-Project/**vroom**
- → https://github.com/DEAP/**deap**

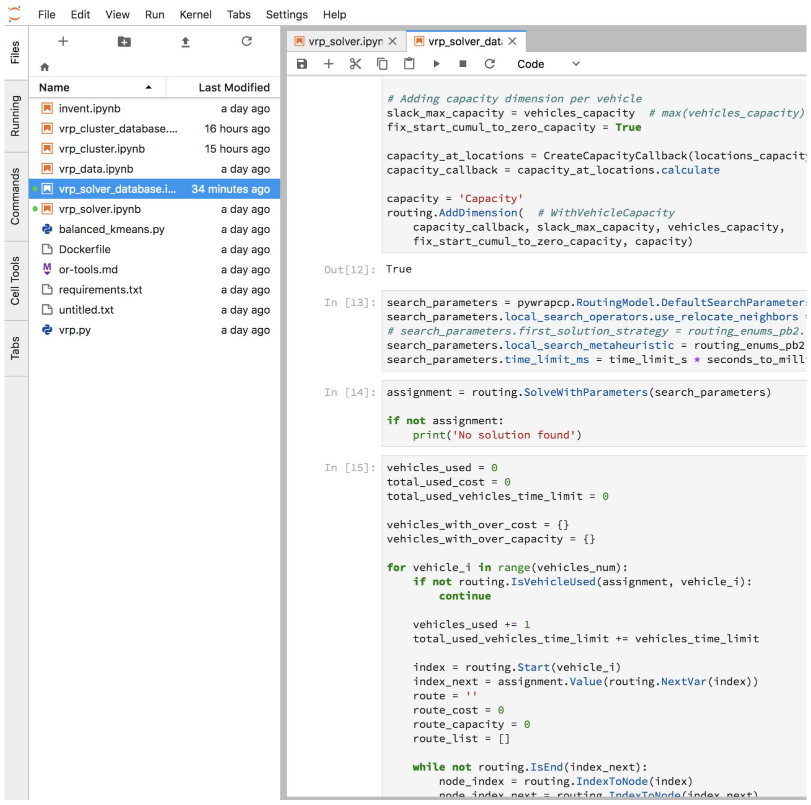
6/

Arquitectura

Workflow inicial

This is the ideal architecture, you may not like it but this is what peak productivity looks like.

```
$> psql
```



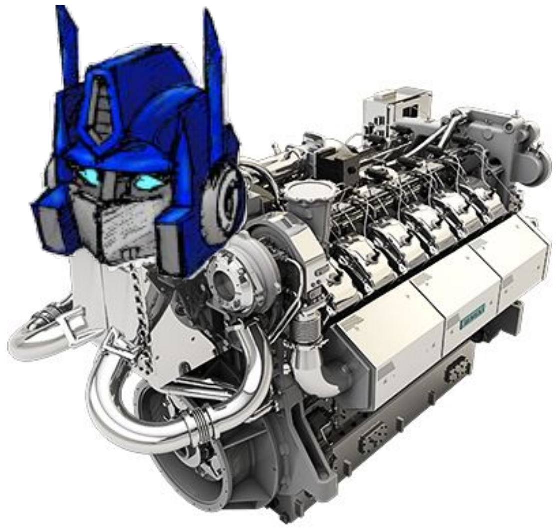
Name	Last Modified
invent.ipynb	a day ago
vrp_cluster_database....	16 hours ago
vrp_cluster.ipynb	15 hours ago
vrp_data.ipynb	a day ago
vrp_solver_database.i...	34 minutes ago
vrp_solver.ipynb	a day ago
balanced_kmeans.py	a day ago
Dockerfile	a day ago
or-tools.md	a day ago
requirements.txt	a day ago
untitled.txt	a day ago
vrp.py	a day ago

```
         # Adding capacity dimension per vehicle
         slack_max_capacity = vehicles_capacity  # max(vehicles_capacity)
         fix_start_cumul_to_zero_capacity = True

         capacity_at_locations = CreateCapacityCallback(locations_capacit
         capacity_callback = capacity_at_locations.calculate

         capacity = 'Capacity'
         routing.AddDimension(  # WithVehicleCapacity
             capacity_callback, slack_max_capacity, vehicles_capacity,
             fix_start_cumul_to_zero_capacity, capacity)

Out[12]: True

In [13]: search_parameters = pywrapcp.RoutingModel.DefaultSearchParameter
         search_parameters.local_search_operators.use_relocate_neighbors
         # search_parameters.first_solution_strategy = routing_enums_pb2.
         search_parameters.local_search_metaheuristic = routing_enums_pb2
         search_parameters.time_limit_ms = time_limit_s * seconds_to_mill

In [14]: assignment = routing.SolveWithParameters(search_parameters)

         if not assignment:
             print('No solution found')

In [15]: vehicles_used = 0
         total_used_cost = 0
         total_used_vehicles_time_limit = 0

         vehicles_with_over_cost = {}
         vehicles_with_over_capacity = {}

         for vehicle_i in range(vehicles_num):
             if not routing.IsVehicleUsed(assignment, vehicle_i):
                 continue

             vehicles_used += 1
             total_used_vehicles_time_limit += vehicles_time_limit

             index = routing.Start(vehicle_i)
             index_next = assignment.Value(routing.NextVar(index))
             route = ''
             route_cost = 0
             route_capacity = 0
             route_list = []

             while not routing.IsEnd(index_next):
                 node_index = routing.IndexToNode(index)
                 node_index_next = routing.IndexToNode(index_next)
```

Arquitectura

Cliente web

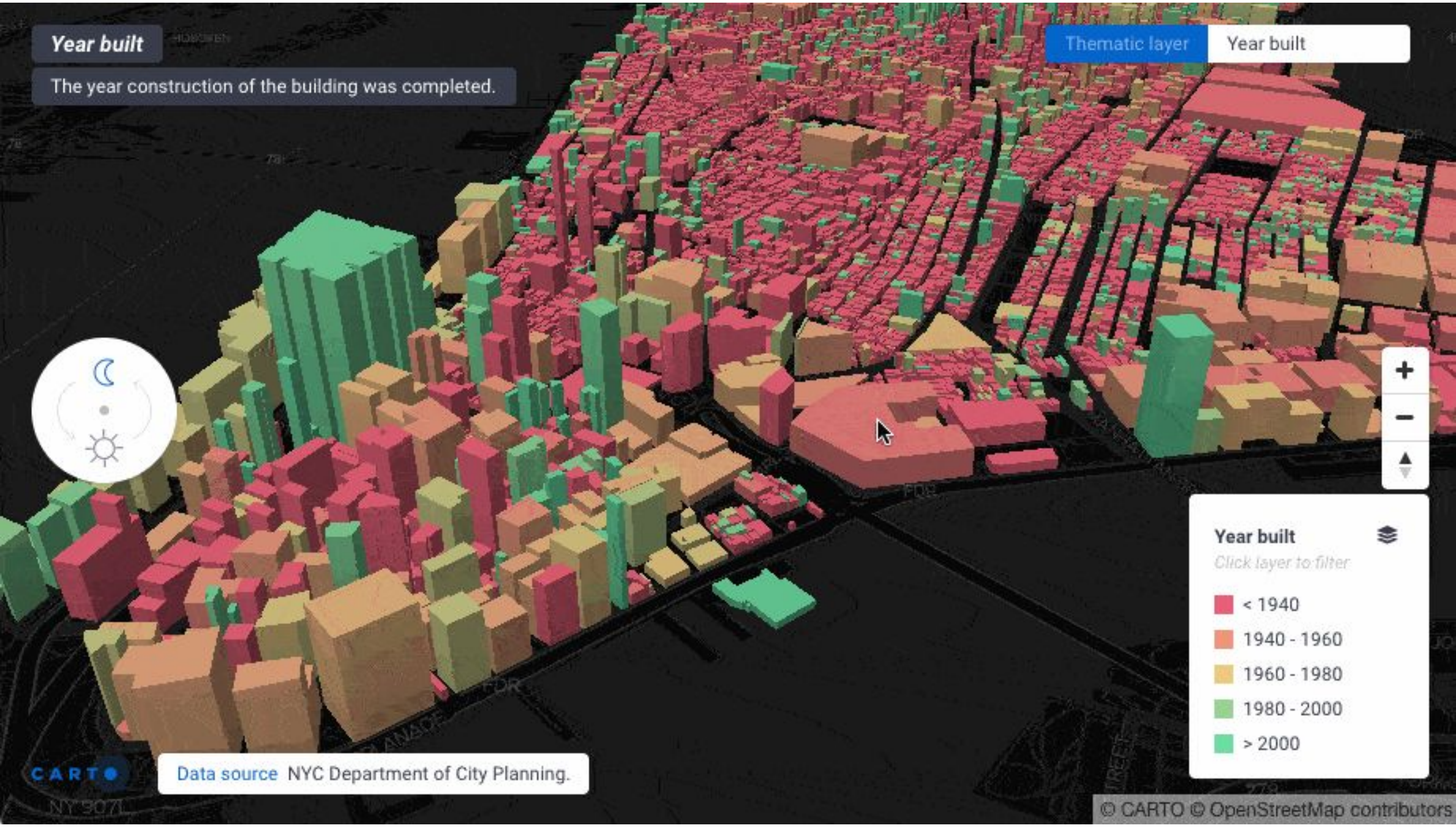
Cliente GIS



API

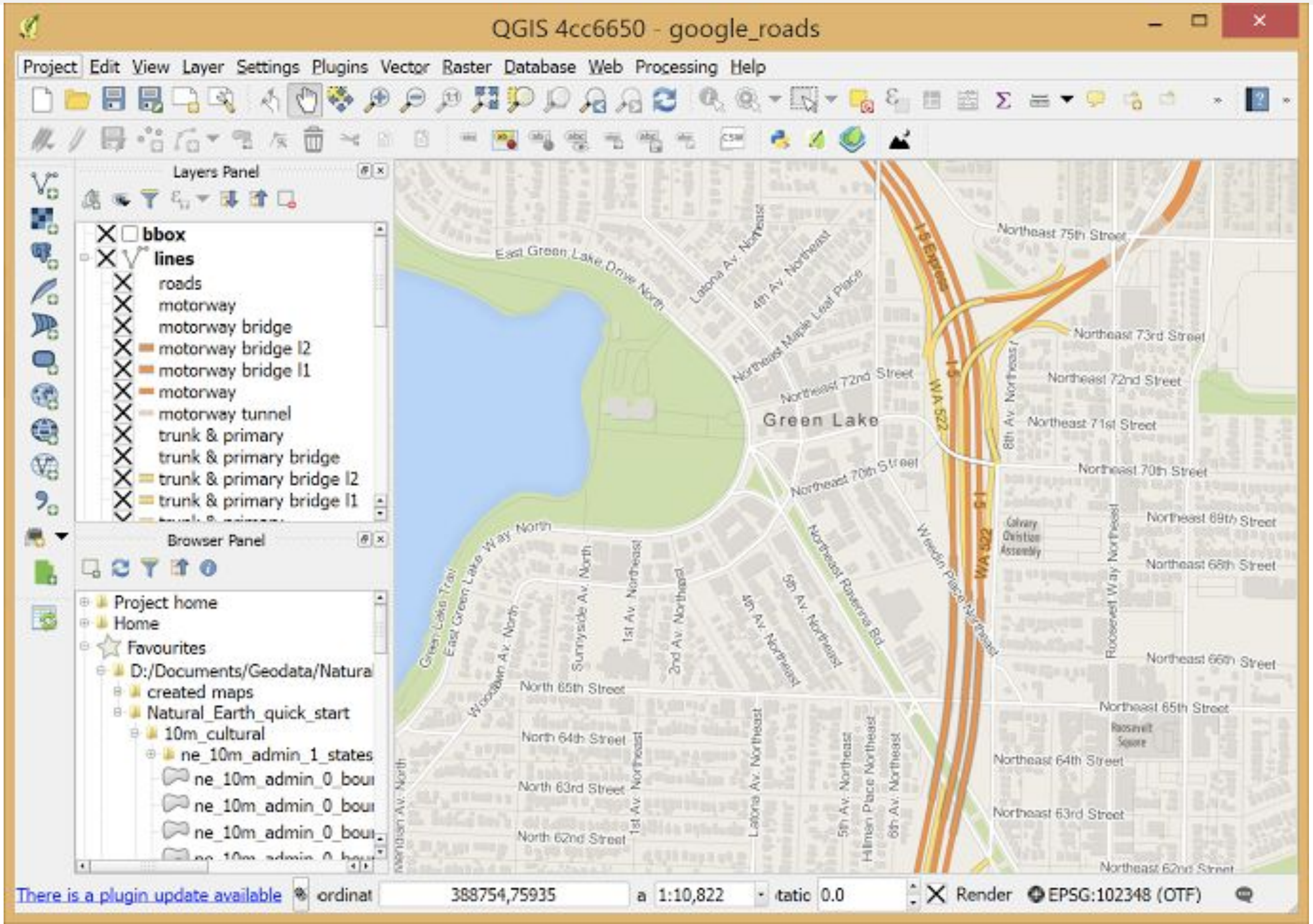
Motor de optimización

Cliente web: CARTO.js

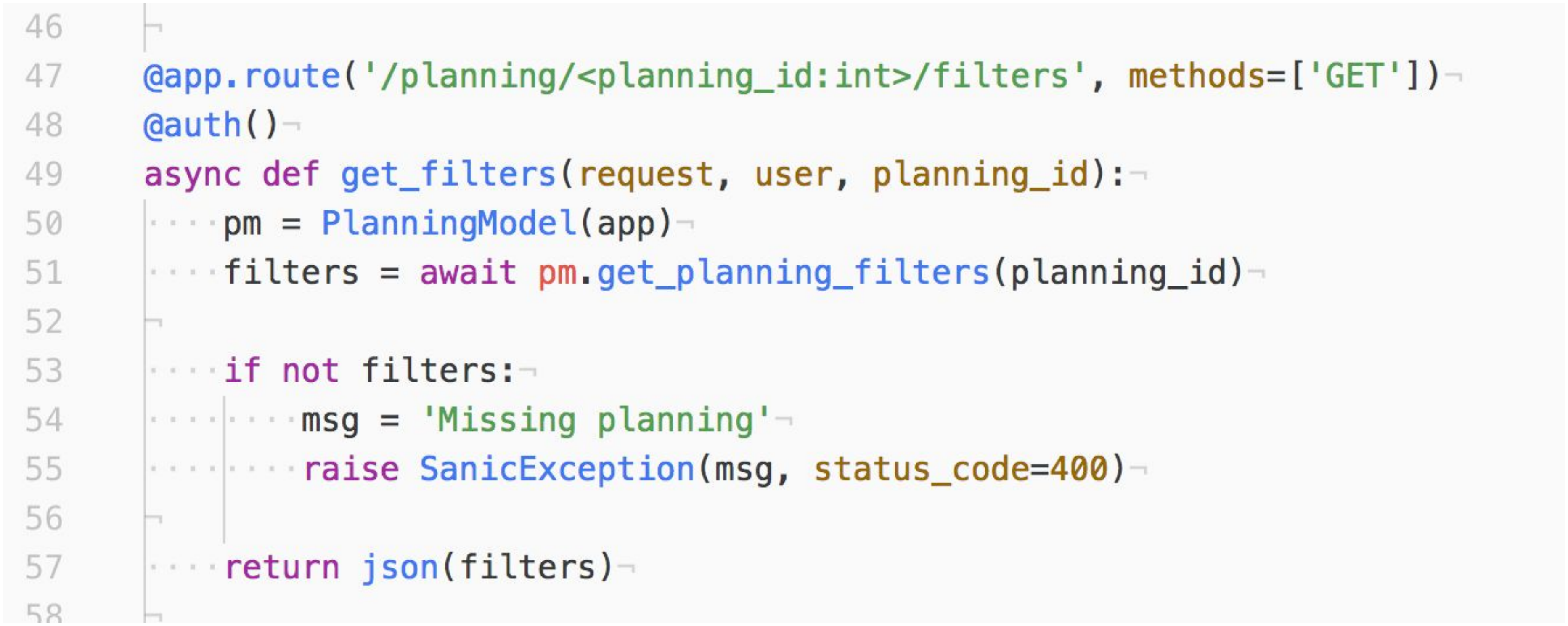


Year built
The year construction of the building was completed.
Thematic layer
Year built
Year built
Click layer to filter
🟥 < 1940
🟧 1940 - 1960
🟨 1960 - 1980
🟩 1980 - 2000
🟩 > 2000
CARTO
Data source NYC Department of City Planning.
© CARTO © OpenStreetMap contributors

Cliente GIS: QGIS + Plugin



API: Sanic



```

@app.route('/planning/<planning_id:int>/filters', methods=['GET'])
@auth()
async def get_filters(request, user, planning_id):
    pm = PlanningModel(app)
    filters = await pm.get_planning_filters(planning_id)

    if not filters:
        msg = 'Missing planning'
        raise SanicException(msg, status_code=400)

    return json(filters)

```

Motor de optimización

pgRouting

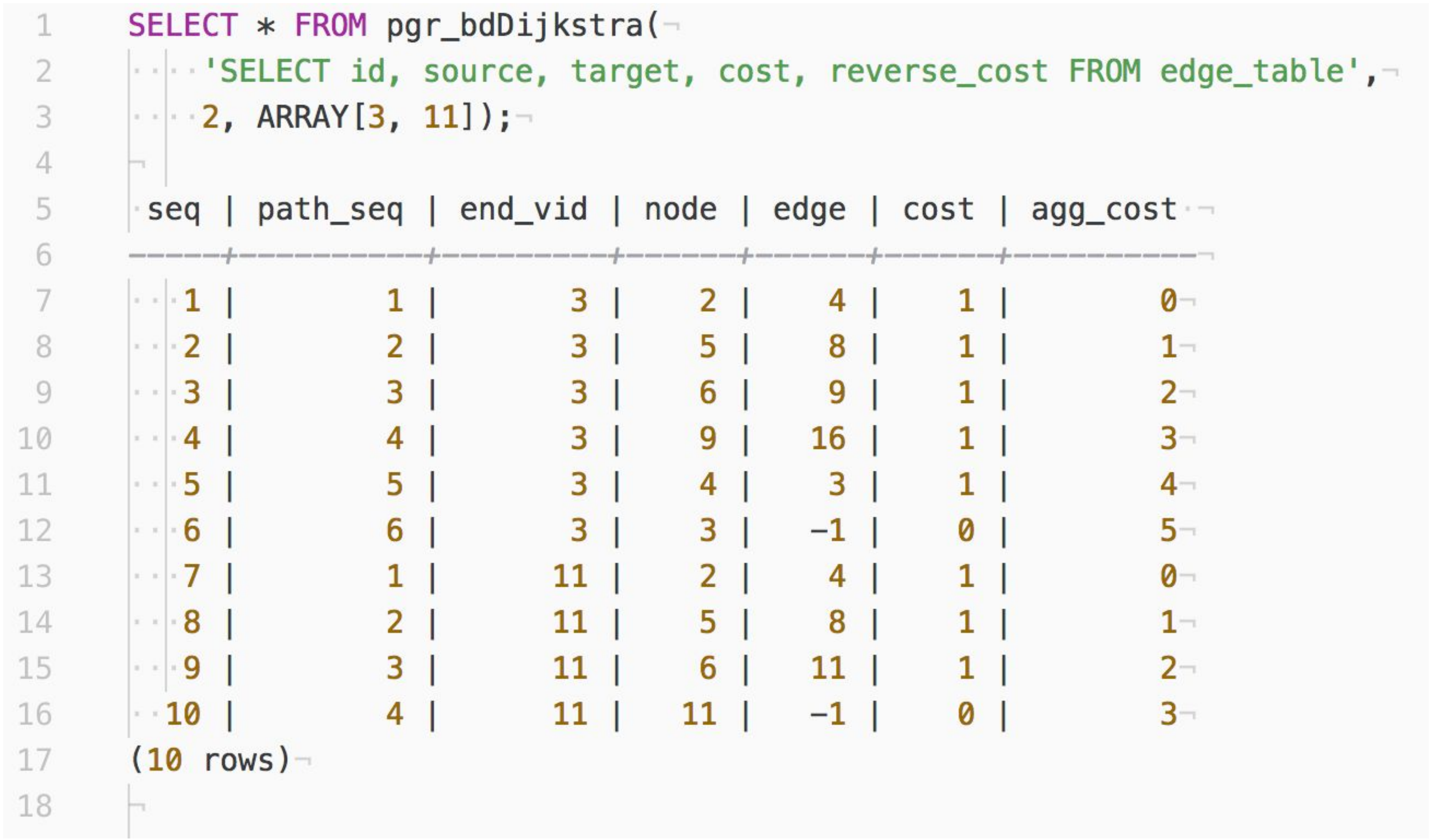
NumPy

pandas

$y_it = \beta' x_{it} + \mu_i + \epsilon_{it}$

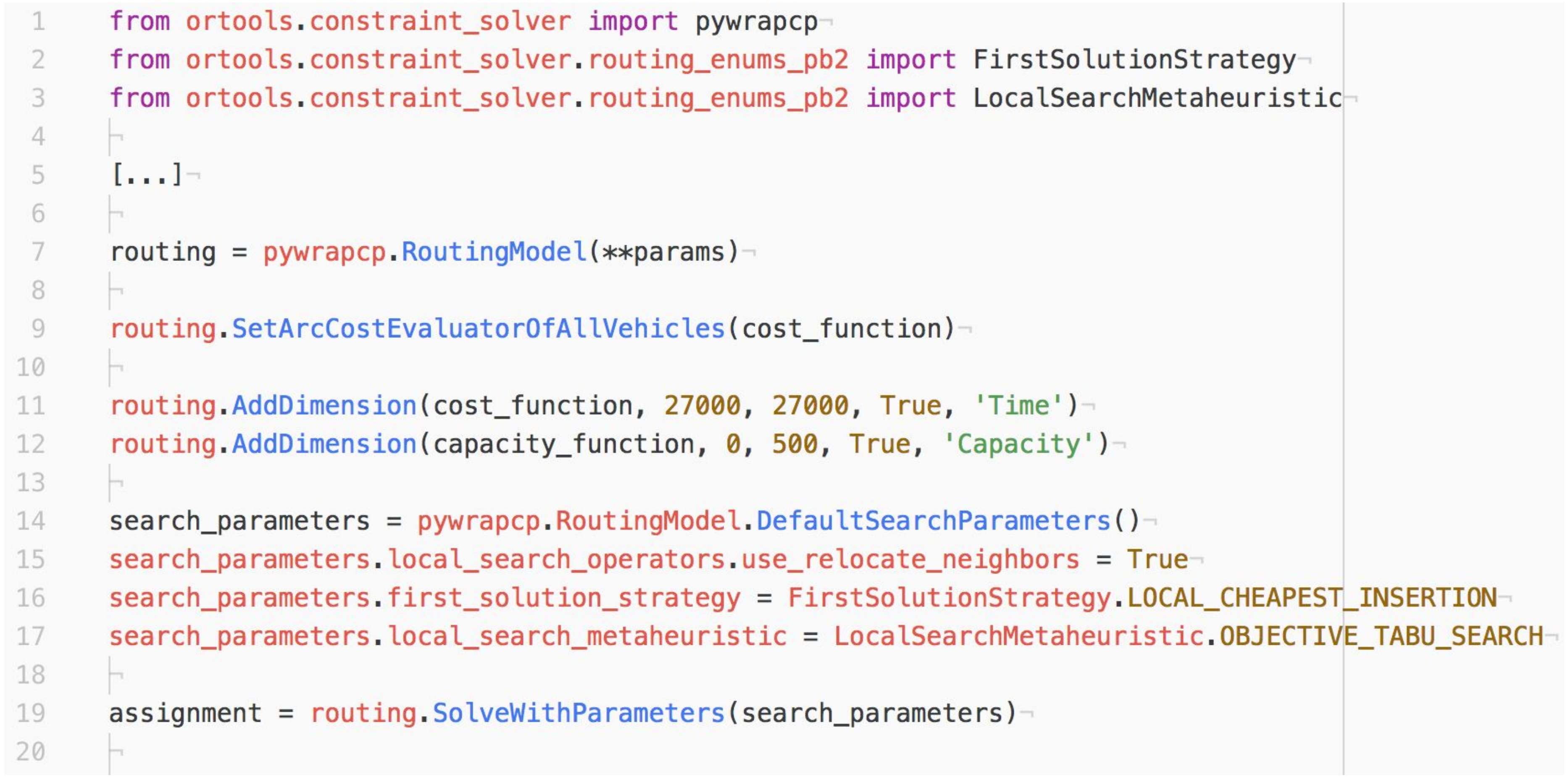
scikit learn

pgRouting



```
SELECT * FROM pgr_bdDijkstra(
    'SELECT id, source, target, cost, reverse_cost FROM edge_table',
    2, ARRAY[3, 11]);

 seq | path_seq | end_vid | node | edge | cost | agg_cost
-----+----------+---------+------+------+------+----------
   1 |        1 |       3 |    2 |    4 |    1 |        0
   2 |        2 |       3 |    5 |    8 |    1 |        1
   3 |        3 |       3 |    6 |    9 |    1 |        2
   4 |        4 |       3 |    9 |   16 |    1 |        3
   5 |        5 |       3 |    4 |    3 |    1 |        4
   6 |        6 |       3 |    3 |   -1 |    0 |        5
   7 |        1 |      11 |    2 |    4 |    1 |        0
   8 |        2 |      11 |    5 |    8 |    1 |        1
   9 |        3 |      11 |    6 |   11 |    1 |        2
  10 |        4 |      11 |   11 |   -1 |    0 |        3
(10 rows)
```

OR-Tools



```python
from ortools.constraint_solver import pywrapcp
from ortools.constraint_solver.routing_enums_pb2 import FirstSolutionStrategy
from ortools.constraint_solver.routing_enums_pb2 import LocalSearchMetaheuristic

[...]

routing = pywrapcp.RoutingModel(**params)

routing.SetArcCostEvaluatorOfAllVehicles(cost_function)

routing.AddDimension(cost_function, 27000, 27000, True, 'Time')
routing.AddDimension(capacity_function, 0, 500, True, 'Capacity')

search_parameters = pywrapcp.RoutingModel.DefaultSearchParameters()
search_parameters.local_search_operators.use_relocate_neighbors = True
search_parameters.first_solution_strategy = FirstSolutionStrategy.LOCAL_CHEAPEST_INSERTION
search_parameters.local_search_metaheuristic = LocalSearchMetaheuristic.OBJECTIVE_TABU_SEARCH

assignment = routing.SolveWithParameters(search_parameters)

```

7/

Siguientes pasos

D

DI s

w

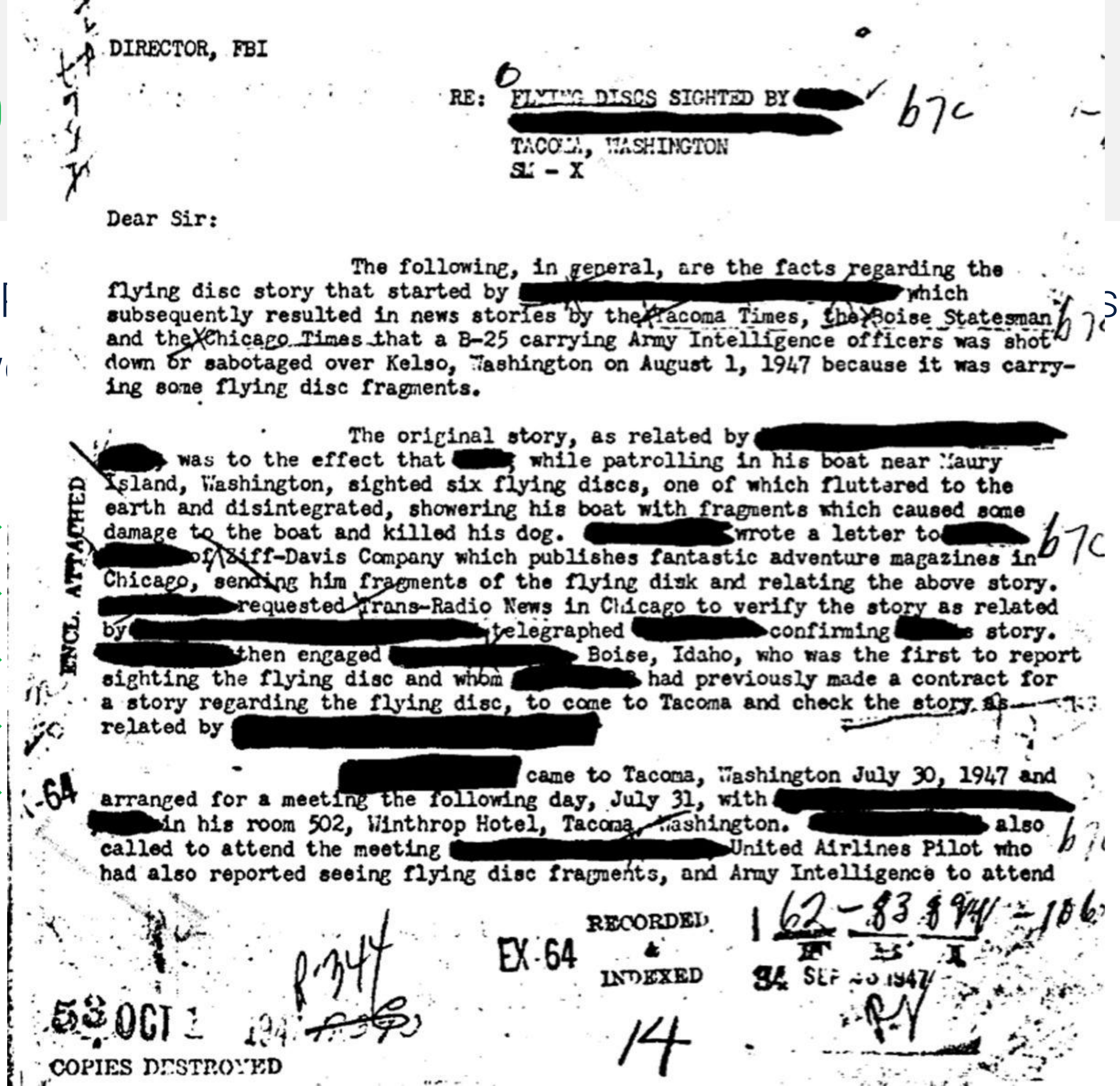
→

→

→

→

→



DIRECTOR, FBI

RE: FLYING DISCS SIGHTED BY [redacted]
[redacted]
TACOMA, WASHINGTON
SM - X

b7c

Dear Sir:

The following, in general, are the facts regarding the flying disc story that started by [redacted] which subsequently resulted in news stories by the Tacoma Times, the Boise Statesman and the Chicago Times that a B-25 carrying Army Intelligence officers was shot down or sabotaged over Kelso, Washington on August 1, 1947 because it was carrying some flying disc fragments. b7c

The original story, as related by [redacted] was to the effect that [redacted] while patrolling in his boat near Maury Island, Washington, sighted six flying discs, one of which fluttered to the earth and disintegrated, showering his boat with fragments which caused some damage to the boat and killed his dog. [redacted] wrote a letter to [redacted] of Ziff-Davis Company which publishes fantastic adventure magazines in Chicago, sending him fragments of the flying disk and relating the above story. [redacted] requested Trans-Radio News in Chicago to verify the story as related by [redacted] telegraphed [redacted] confirming [redacted]s story. [redacted] then engaged [redacted] Boise, Idaho, who was the first to report sighting the flying disc and whom [redacted] had previously made a contract for a story regarding the flying disc, to come to Tacoma and check the story as related by [redacted] b7c

ENCL. ATTACHED

[redacted] came to Tacoma, Washington July 30, 1947 and arranged for a meeting the following day, July 31, with [redacted] in his room 502, Winthrop Hotel, Tacoma, Washington. [redacted] also called to attend the meeting [redacted] United Airlines Pilot who had also reported seeing flying disc fragments, and Army Intelligence to attend b7c

-64

EX-64 RECORDED & INDEXED 62-83894-186

34 SEP 1947

R344

53 OCT 1 194

14

COPIES DESTROYED

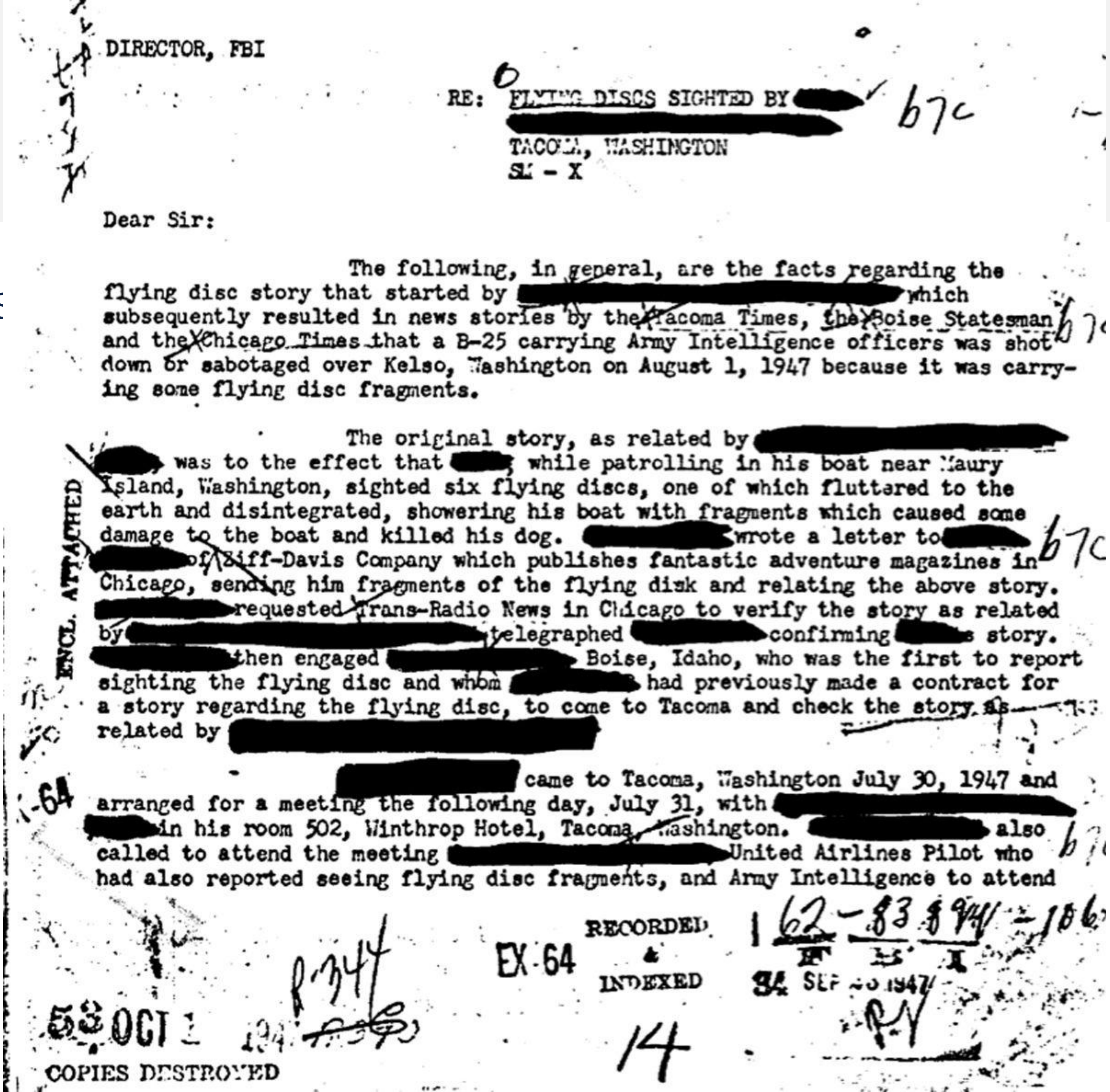
D

Es

→

→

→



DIRECTOR, FBI

RE: FLYING DISCS SIGHTED BY [redacted] b7c
[redacted]
TACOMA, WASHINGTON
SM - X

Dear Sir:

The following, in general, are the facts regarding the flying disc story that started by [redacted] which subsequently resulted in news stories by the Tacoma Times, the Boise Statesman and the Chicago Times that a B-25 carrying Army Intelligence officers was shot down or sabotaged over Kelso, Washington on August 1, 1947 because it was carrying some flying disc fragments. b7c

The original story, as related by [redacted] was to the effect that [redacted] while patrolling in his boat near Maury Island, Washington, sighted six flying discs, one of which fluttered to the earth and disintegrated, showering his boat with fragments which caused some damage to the boat and killed his dog. [redacted] wrote a letter to [redacted] of Ziff-Davis Company which publishes fantastic adventure magazines in Chicago, sending him fragments of the flying disk and relating the above story. [redacted] requested Trans-Radio News in Chicago to verify the story as related by [redacted] telegraphed [redacted] confirming [redacted]s story. [redacted] then engaged [redacted] Boise, Idaho, who was the first to report sighting the flying disc and whom [redacted] had previously made a contract for a story regarding the flying disc, to come to Tacoma and check the story as related by [redacted] b7c

[redacted] came to Tacoma, Washington July 30, 1947 and arranged for a meeting the following day, July 31, with [redacted] in his room 502, Winthrop Hotel, Tacoma, Washington. [redacted] also called to attend the meeting [redacted] United Airlines Pilot who had also reported seeing flying disc fragments, and Army Intelligence to attend b7c

ENCL. ATTACHED

X-64

R 344

EX-64

RECORDED & INDEXED

62-83894-106

34 SEP 1947

14

53 OCT 1 194

COPIES DESTROYED

Alberto Asuero
Chief Technology Officer
@alasarr
alberto@geographica.gs

Cayetano Benavent
Head of Data
@cayetanobv
cayetano@geographica.gs

Josema Camacho
Chief Innovation Officer
@josemazo
josema@geographica.gs

Thanks.

www.geographica.gs

G E O
G R A
P H I
C A .