# ATLÂNTICOS

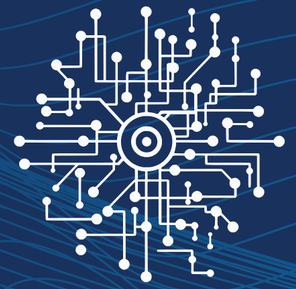
# AEGEA

ATLÂNTICOS

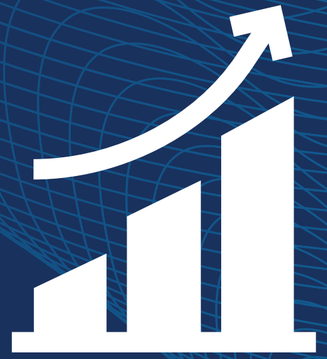
# ROAD MAP

Recaptulação do problema

Contextualização da soulução



Algoritmos



Resultadoo

ATLÂNTICOS

# CONTEXTO DO PROBLEMA

- **Desafio: A Aegea Saneamento busca otimizar as rotas dos leituristas na concessionária Águas do Rio, no Rio de Janeiro.**

- **Objetivo: Melhorar a precisão do faturamento, reduzir custos e aumentar a eficiência operacional.**

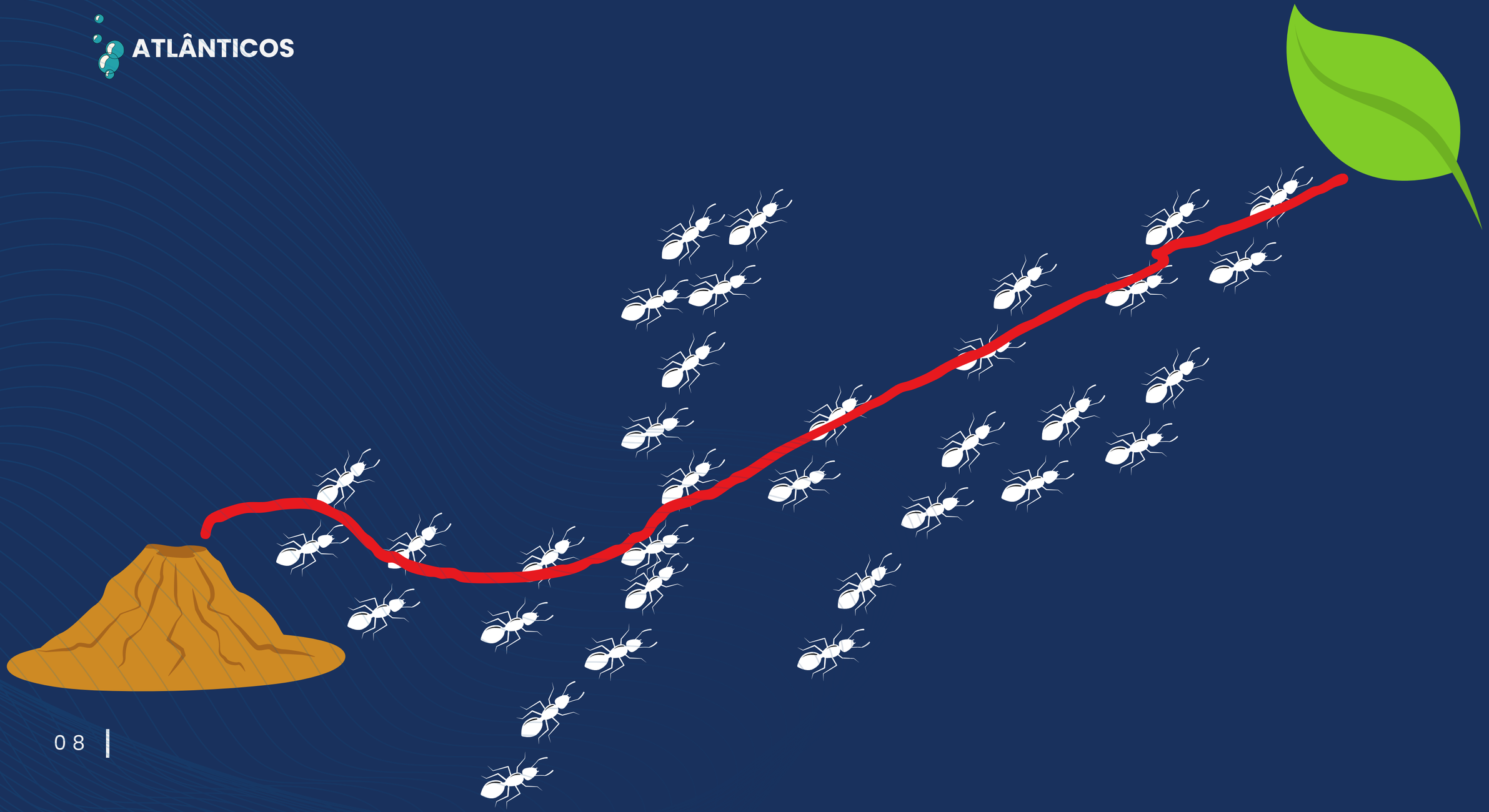
- **Importância: Otimizar as rotas pode impactar diretamente a eficiência das leituras, a entrega de contas e o custo operacional para a empresa.**

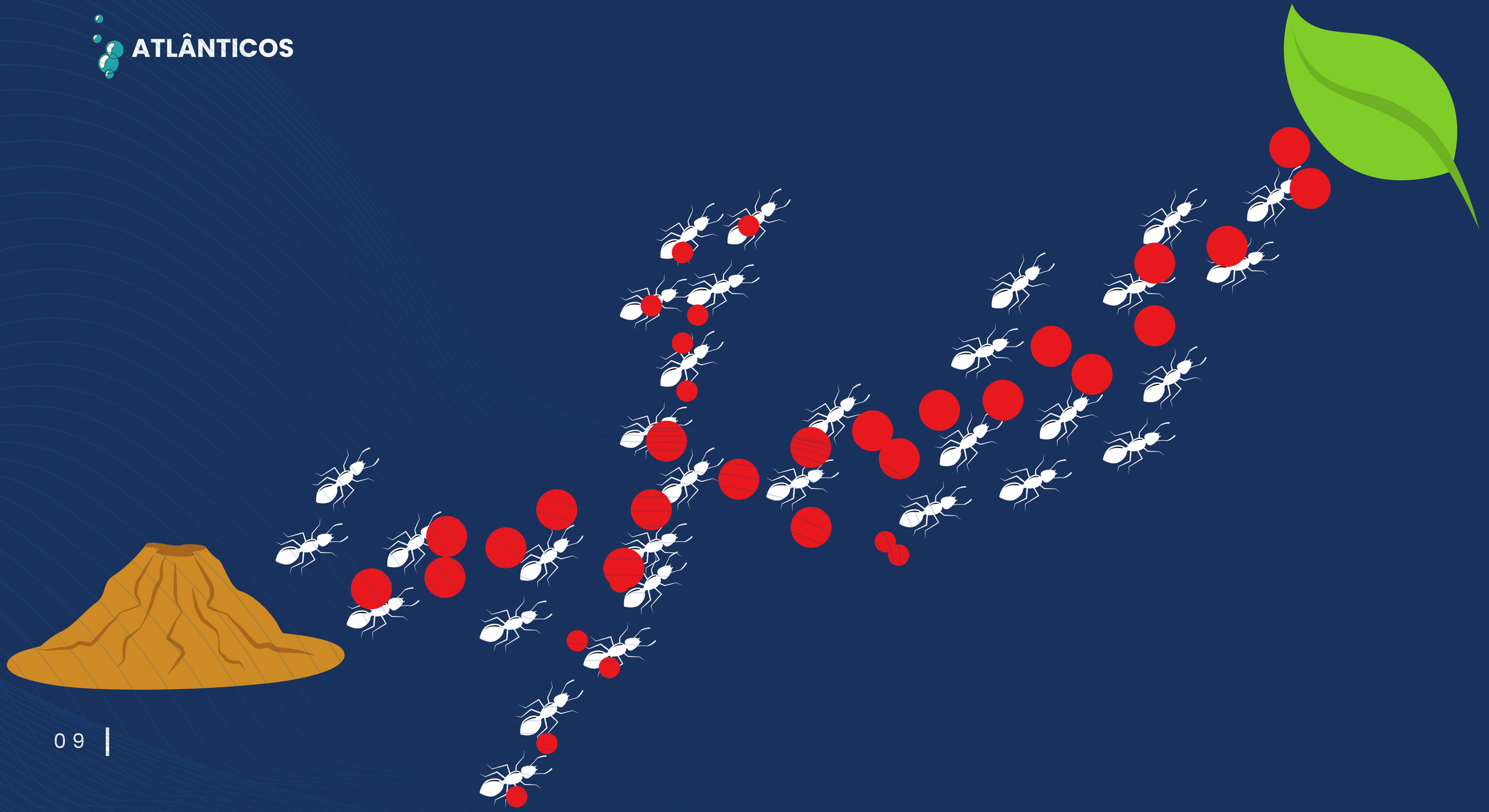
ATLÂNTICOS

# COMO PODEMOS RESOLVER ESSE PROBLEMA?

ATLÂNTICOS

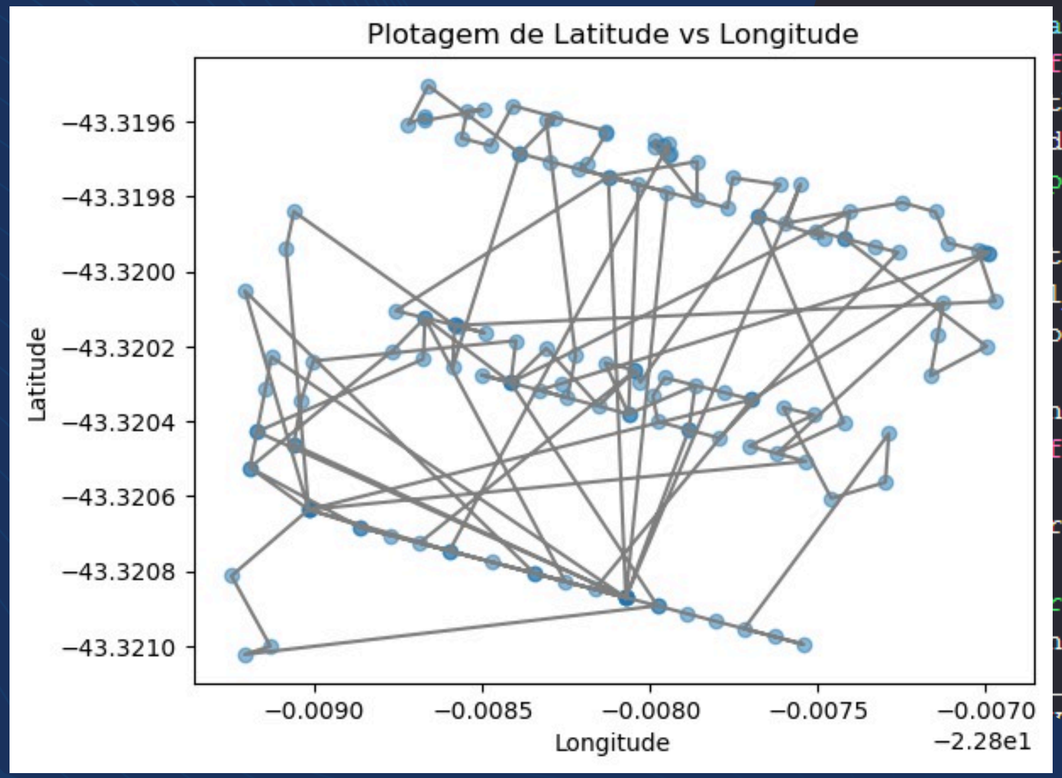
# FORMIGAS





ATLÂNTICOS

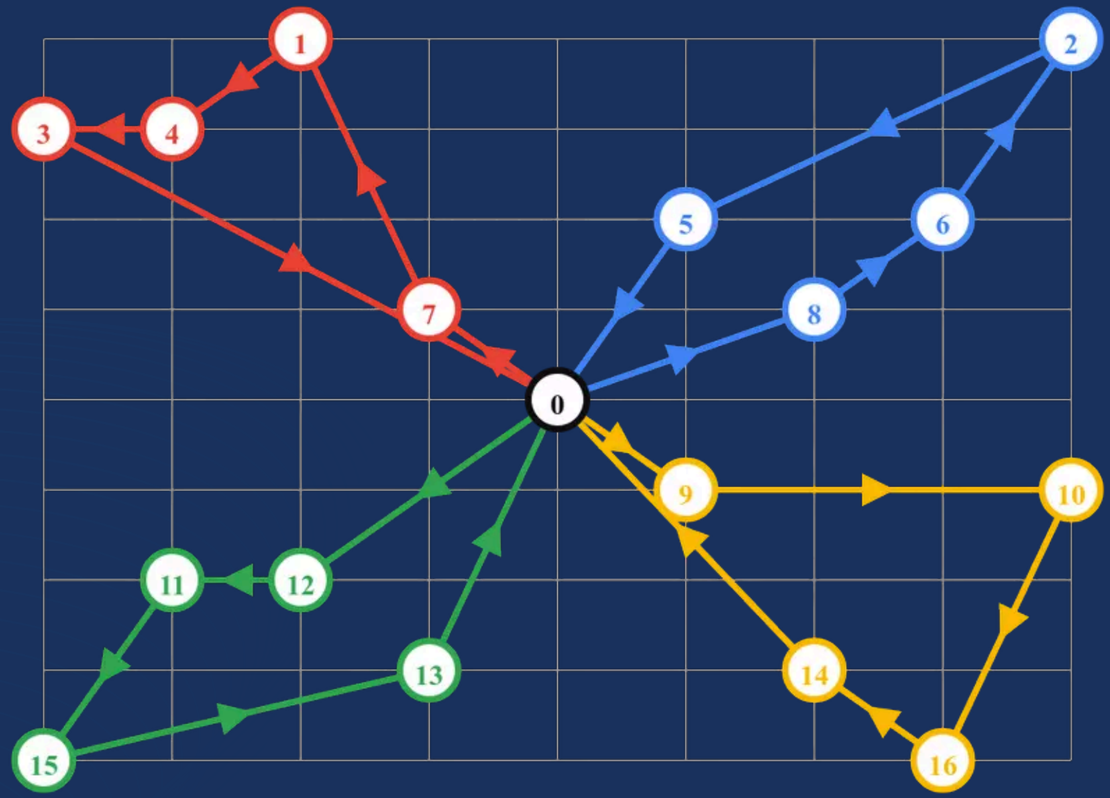
# ANT-COLONY OPTIMIZATION




```
                distances
        = np.ones(self.distances.shape) / len(dista
    ds = range(len(distances))
    ts = n_ants
    best = n_best
    n_iterations = n_iterations
    .decay = decay
    lf.alpha = alpha
    elf.beta = beta

run(self):
shortest_path = []  # Initialize shortest_path as an empty list
all_time_shortest_path = (None, np.inf)

    ange(self.n_iterations):
    f"Iteration {i+1}/{self.n_iterations}")
    ths = self.gen_all_paths()
    dist = min(all_paths, key=lambda x: x[1])
    pdate_pheromone(all_paths, shortest_path)

    t < all_time_shortest_path[1]:
    l_time_shortest_path = (path, dist)
    ortest_path = path

    heromone *= self.decay
    f"Current shortest path: {shortest_path}")

    rtest_path, all_time_shortest_path[1]

    romone(self, all_paths, shortest_path):
    ns = sorted(all_paths, key=lambda x: x[1])
    _ in sorted_paths[:self.n_best]:
    ve in path:
            romone[move] += 1.0 / self.distances[m
```

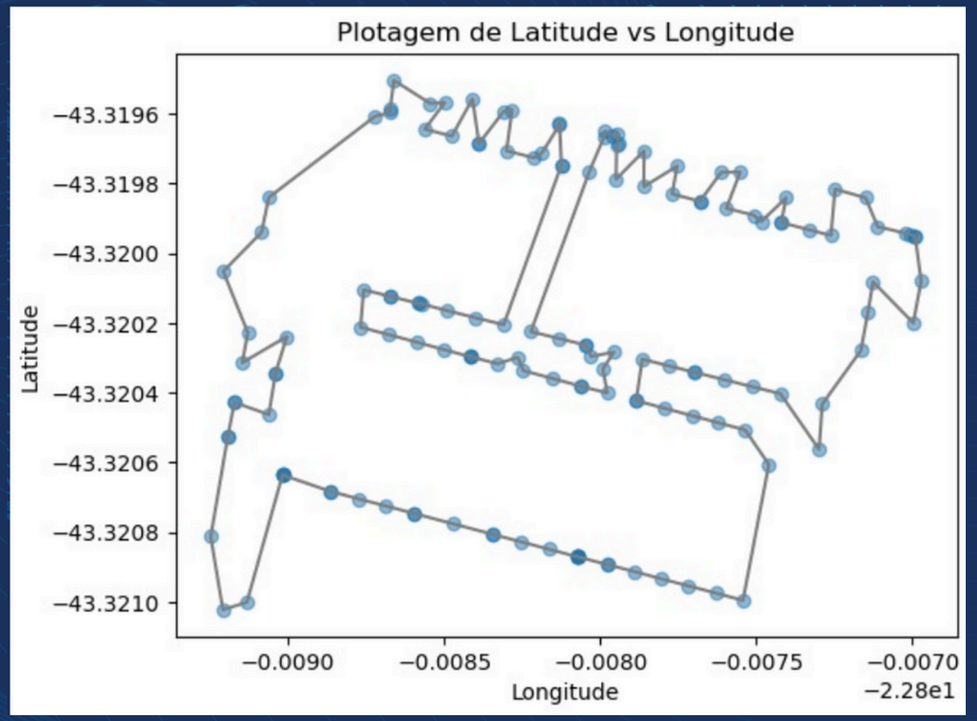
# OR TOOLS

**OR-Tools é um conjunto de bibliotecas e ferramentas desenvolvidas pelo Google**



ATLÂNTICOS

# OR TOOLS

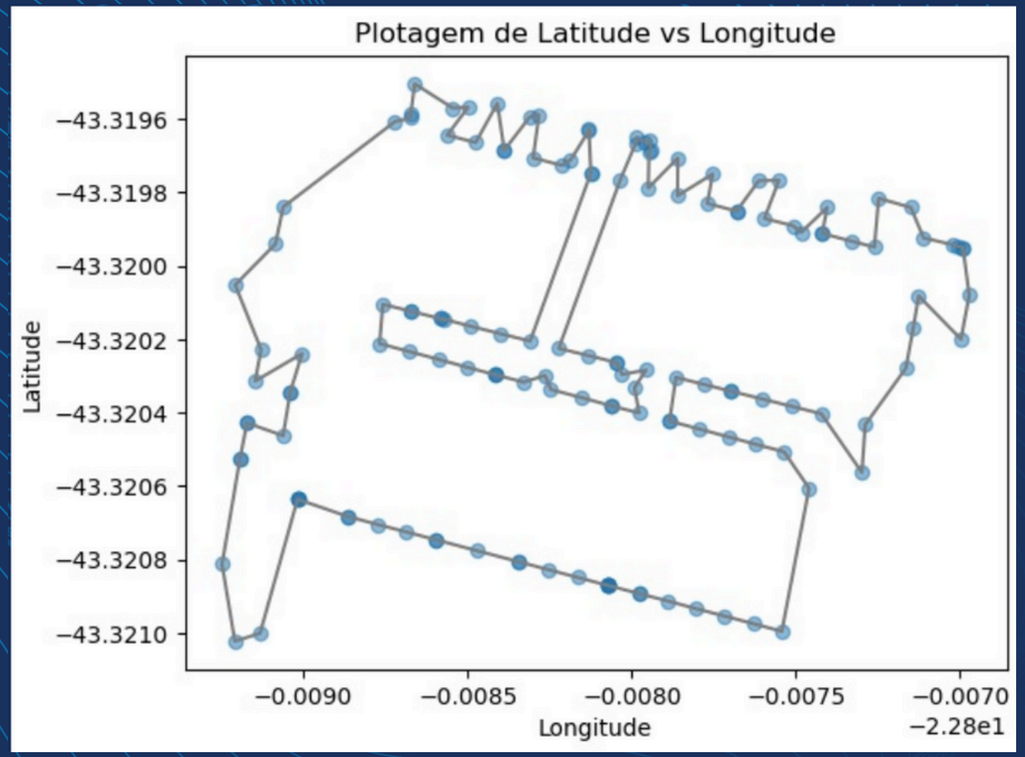



```
                                    quivo CSV
                                 data/AMOTRA_MENOR.csv', sep=';')

                                  DataFrame
                                data.copy()

                                s coordenadas de latitude e longitude para metros usando
                              ed_data['LATITUDE'].values * 10000000 / 90  # 90 graus == 10.
                            cluestered_data['LATITUDE'].values * np.pi / 180.0)
                           ered_data['LONGITUDE'].values * k * 10000000 / 90  # 90 graus ==

                           dos valores mínimos
                          np.min(x)
                          np.min(y)

                          ce dos valores relativos ao mínimo encontrado
                         os = (x - minx1).astype(int)
                         os = (y - minx2).astype(int)

                         ção de colunas ao DataFrame
                        ered_data['x_metros'] = x_metros
                        ered_data['y_metros'] = y_metros

                        ição do número de clusters
                        s = 30

                             dos dados usando KMeans
                           clusters=n_clusters, random_state=42, n_init=1
                              data[['x_metros', 'y_metros']])
                                kmeans.labels_
```

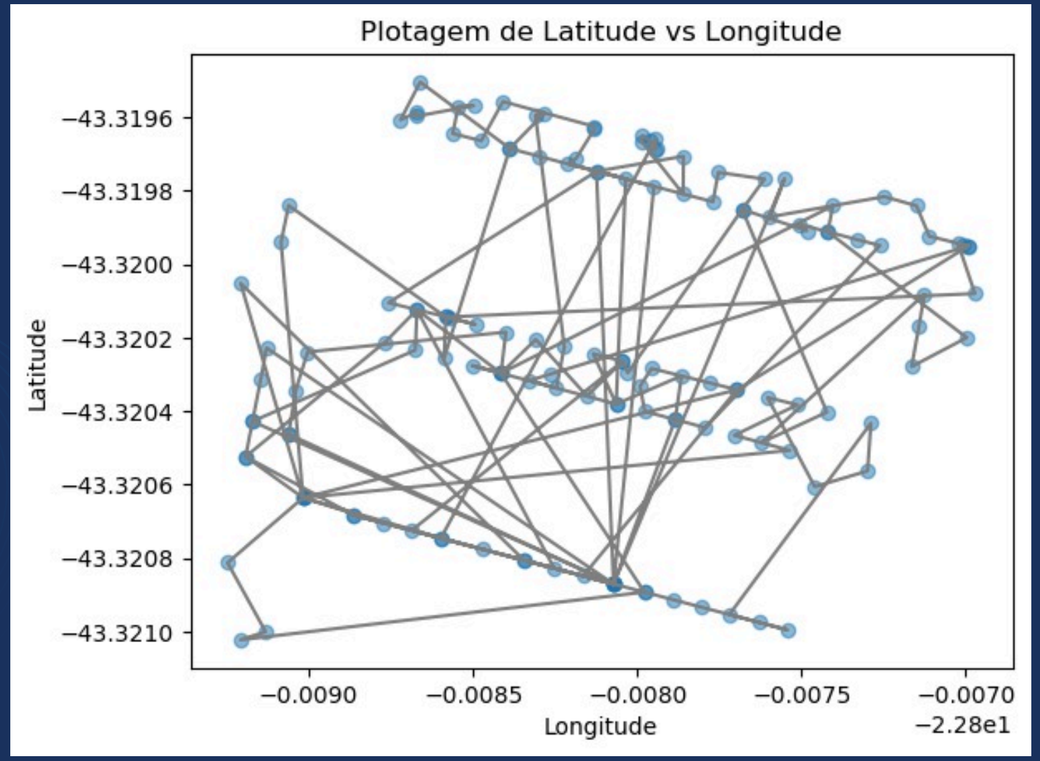
ATLÂNTICOS

# Algoritmos

## OR Tools




## Antcolony



ATLÂNTICOS

# ATLANTICOS TEAM

KATSUKI

FELIPE

FERNANDO

GABRIEL

LEONARDO

ANDRÉ

GABRIEL

# OBRIGADO

pela atenção