Routing Algorithm for Ocean Shipping and Urban Deliveries

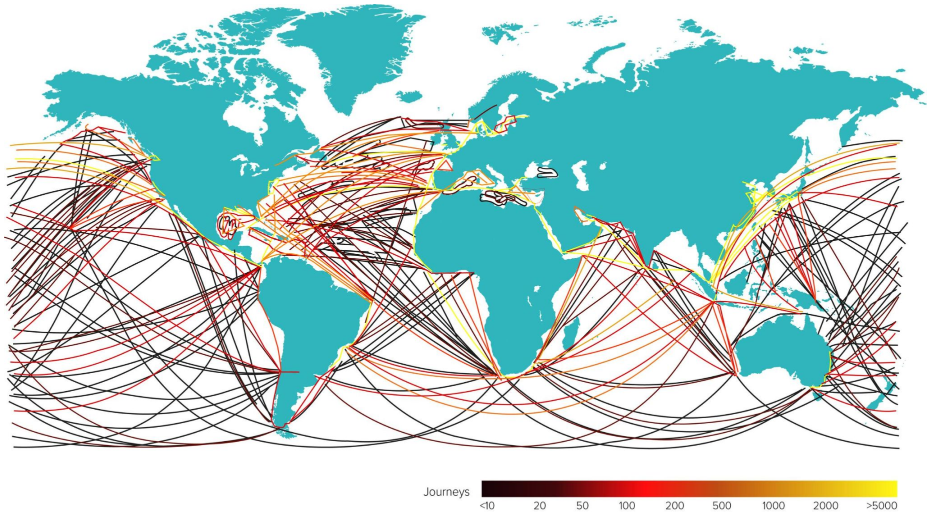
Grupo 5 2LEIC19

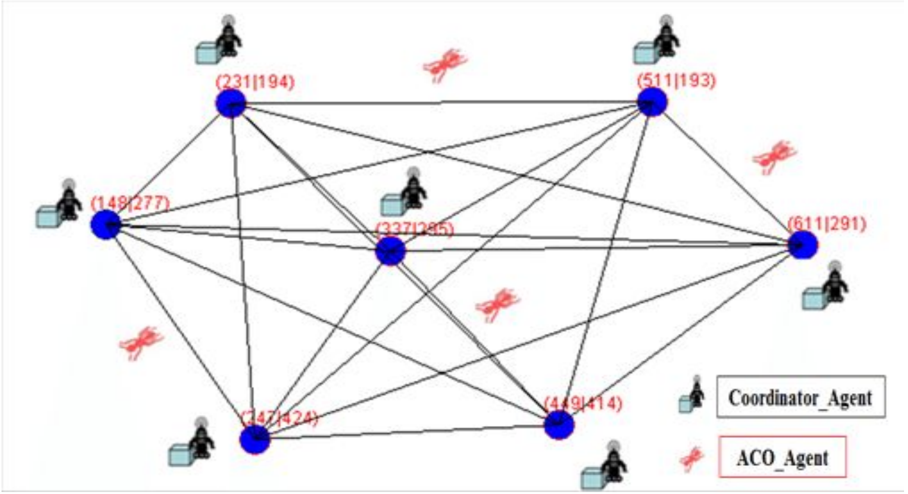
- *Afonso Castro* up202208026
- *Pedro Santos* up202205900
- *Rodrigo de Sousa* up202205751

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Graph

- Graphs are invaluable tools for illustrating interconnected systems.
- Here, we employ a graphical representation to delineate the intricate network of optimal routes for vehicles in generic shipping and delivery scenarios, from urban deliveries to ocean shipping.
- We use the same Data Structure for this graph as the one that was provided in class, with no changes made. We created a class Node (mentioned later) to implement the changes needed.





Read Functions

static Graph<Node> readExtra(int i);

static Graph<Node> readReal(int i);

```cpp
Graph<Node> ReadFunctions::readExtra(int i) {
    cout << string(LINE_SIZE_, '-') << endl;
    cout << "Loading data contents..." << endl;
    cout << string(LINE_SIZE_, '-') << endl;

    vector<Node> nodes_extra;
    Graph<Node> g;
    if (i != 25 && i != 50 && i != 75 && i != 100 && i != 200 && i != 300 && i != 400 && i != 500 && i != 600 && i != 700 && i != 800 && i != 900) { cout << "Error: Extra csv number doesn't exist."; return g;}

    string path = "../csv/extra/edges_" + to_string(i) + ".csv";
    string nodepath = "../csv/extra/nodes.csv";


    ifstream file(nodepath);
    if (!file.is_open()) {cerr << "Error (204): Wrong Path"; return g;}
    string fLine;
    getline(file, fLine);

    string line;
    while (getline(file, line)) {
        istringstream ss(line);
        string id1, lon, lat;
        if (getline(ss, id1, ',') && getline(ss, lon, ',') && getline(ss, lat, ',')) {
            pair<double, double> coordinates = {stod(lon),stod(lat)};
            Node a(stoi(id1),"N/A",coordinates);
            if (g.findVertex(a)==nullptr){g.addVertex(a); a.print();}
        }
    }


    ifstream file2(path);
    if (!file2.is_open()) {cerr << "Error (204): Wrong Path"; return g;}
    string fLine2;
    getline(file, fLine2);


    string line2;
    while (getline(file2, line2)) {
        istringstream ss(line2);
        string id1, id2, distance;
        if (getline(ss, id1, ',') && getline(ss, id2, ',') && getline(ss, distance, ',')) {

            Node a(stoi(id1),"N/A", {0.0,0.0});
            Node z(stoi(id2), "N/A", {0.0,0.0});
            g.addEdge(g.findVertex(a)->getInfo(), g.findVertex(z)->getInfo(), stod(distance));

            g.addEdge(g.findVertex(z)->getInfo(), g.findVertex(a)->getInfo(), stod(distance));
        }
    }

    return g;
}
```

Read Functions

static Graph<Node> readExtra(int i);

static Graph<Node> readReal(int i);

```
Graph<Node> ReadFunctions::readReal(int i) {
    cout << string(LINE_SIZE_, '-') << endl;
    cout << "Loading data contents..." << endl;
    cout << string(LINE_SIZE_, '-') << endl;

    vector<Node> nodes_real;
    Graph<Node> g;
    if (i < 1 && i > 3) { cout << "Error: Real Graph number doesn't exist."; return g;}

    string path = "../csv/real/graph" + to_string(i) + "/";
    string nodepath = path + "nodes.csv";
    string edgepath = path + "edges.csv";


    ifstream file(nodepath);
    if (!file.is_open()) {cerr << "Error (204): Wrong Path"; return g;}
    string fLine;
    getline(file, fLine);

    string line;
    while (getline(file, line)) {
        istringstream ss(line);
        string id1, lon, lat;
        if (getline(ss, id1, ',') && getline(ss, lon, ',') && getline(ss, lat, ',')) {
            pair<double, double> coordinates = {stod(lon),stod(lat)};
            Node a(stoi(id1),"N/A",coordinates);
            if (g.findVertex(a)==nullptr){g.addVertex(a); a.print();}
        }
    }


    ifstream file2(edgepath);
    if (!file2.is_open()) {cerr << "Error (204): Wrong Path"; return g;}
    string fLine2;
    getline(file2, fLine2);


    string line2;
    while (getline(file2, line2)) {
        istringstream ss(line2);
        string id1, id2, distance;
        if (getline(ss, id1, ',') && getline(ss, id2, ',') && getline(ss, distance, ',')) {

            Node a(stoi(id1),"N/A", {0.0,0.0});
            Node z(stoi(id2), "N/A", {0.0,0.0});
            g.addEdge(g.findVertex(a)->getInfo(), g.findVertex(z)->getInfo(), stod(distance));
        }
    }

    return g;
}
```

Operation Functions

- static double **distance**(Graph<Node> &graph, Node src, Node dest);
- static void **backtracking**(Graph<Node> &graph, vector<int> &path, vector<int> &minpath, Node current_city, double &min_distance, double &total_distance);
- static void **bound_2**(Graph<Node> &graph);
- static vector<Vertex<Node> *> **prims**(Graph<Node> &graph, int i);
- static Graph<Node> **primsGraph**(Graph<Node> &graph, int i);
- static void **tApprox**(Graph<Node> &graph);
- static Vertex<Node> ***getVertexRealWorldCoordinates**(Graph<Node> &graph, double lat, double lon);
- static void **christofides**(Graph<Node> &graph, int start, bool real);

Data Selection Menu

```
-----------------------------------------------------------------------------------------------
Welcome to the Ocean Shipping and Urban Deliveries Routing Algorithm Tool!
-----------------------------------------------------------------------------------------------
Select your data folder:
1 - Toy.
2 - Extra Medium.
3 - Real-world.
0 - Exit.
-----------------------------------------------------------------------------------------------
Please select the task you wish to perform by inputting its number:
```

Operations Menu

```
Select your operation:
----------------------------------------------------------------------------------------
1 - Backtracking Algorithm.
2 - Triangular Approximation Heuristic.
3 - Alternative Heuristic.
9 - Credits.
0 - Exit.
----------------------------------------------------------------------------------------
Please select the task you wish to perform by inputting its number:
```

Starting Point Menu

```
------------------------------------------------------------------------------------------
Select Starting Point:
------------------------------------------------------------------------------------------
1 - Random.
2 - ID.
3 - Coordinates.
0 - Exit.
------------------------------------------------------------------------------------------
Please select the task you wish to perform by inputting its number: 

Please enter the Longitude coordinate of the desired starting point: -27
Please enter the Latitude coordinate of the desired starting point: 42
```

Node

```
class Node {
    private:
        int index_; //number of the index of the node
        string label_; //info related to the node (such as the labels in tourism.csv)
        pair<double,double> coordinates_;
```

Time Complexity

Time Complexities for each method:

- **Backtracking Algorithm:** aprox. $O(n!)$
- **Triangular Approximation Algorithm:** aprox. $O(E \log V)$
- **Christofides Algorithm:** aprox. $O(n^2)$

Algorithms' Analysis

Backtracking:

- **Less Efficiency:** Useless in Large Graphs;
- **Higher Precision:** Results are the Most Accurate;

Algorithms' Analysis

Triangular Approximation:

- **Higher Efficiency:** Almost Instant;
- **Low Precision:** An Approximation;

Algorithms' Analysis

Christofides:

- **Best of Both Worlds;**
- **Lower Complexity:** Only higher because of the minimum weight perfect matching algorithm;
- **Higher Precision:** Delivers the closest result to the Backtracking Algorithm.

Highlights and Difficulties

- In this project, the Backtracking and the Triangular Approximation Algorithms were applied with relative ease.
- On the other hand, the quest for the Alternative Heuristic was in our opinion the most challenging part of the project. We tried a couple of methods until we ended up applying the Christofides Algorithm, which was without a doubt the most complicated part of the code, as we also wanted to make it work for the real world graphs.

FIM

Equal participation from all members of the group.