Routing Algorithm for Ocean Shipping and Urban Deliveries

Filipe Correia
Gabriela Silva
Gonçalo Nunes
Grupo G12_2

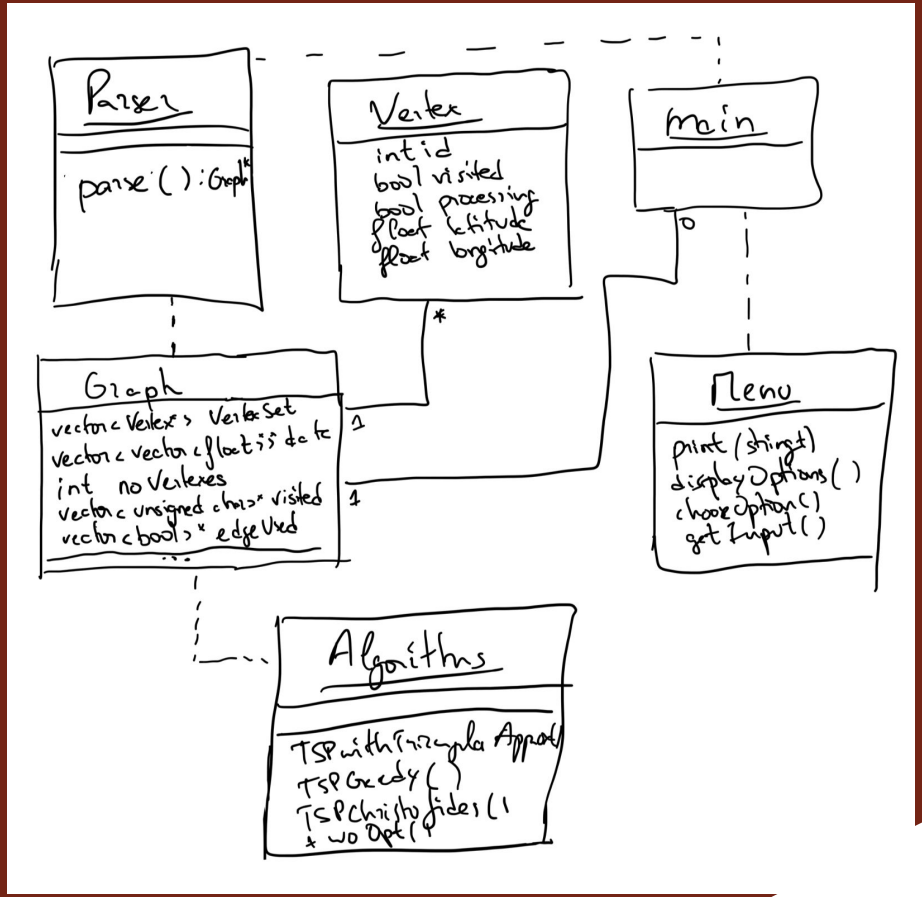
U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

O que implementámos?

- Parser
- Menu Intuitivo
- Documentação
- Algoritmo de *Backtracking*
- Triangular Approximation Algorithm
- Diversos algoritmos aplicáveis para os grafos completamente ligados
- Algoritmo Greedy Adaptado aplicável para o grafo “Real World”
- 2-opt Heuristic

Finalmente, as funções mais relevantes, contêm informação sobre complexidade temporal.

Visão Geral da Estrutura



class Graph

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

class Graph

```
class Graph{
    std::vector<Vertex*> vertexSet;    // vertex set
    std::vector<std::vector<float>*>* data; // save
it as negative if not in graph - if 0, it has not
been calculated yet
    int noVertexes;
    std::vector<unsigned char>* visited;
    std::vector<bool>* edgeUsed;
}
```

```
class Vertex{
    int id;
    bool visited = false;
    bool processing = false;
    float latitude;
    float longitude;
    int nextVertex=-1;
}
```

Algoritmos

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Backtracking - O(V!)

- Um algoritmo simples de backtracking recursivo para o problema de TSP.
- As chamadas recursivas são realizadas com o custo mínimo encontrado até ao momento como argumento. Se o valor atual já for superior ao do custo mínimo, sabendo que os pesos das edges correspondem a distâncias reais e, portanto, não existem pesos negativos, é realizado o pruning do ramo atual por ser impossível de nos levar a uma solução ótima.
- Apesar deste algoritmo conduzir sempre a uma solução ótima, é muito custoso em tempo e, por isso, infeasible para grafos maiores

Triangular Approximation Algorithm

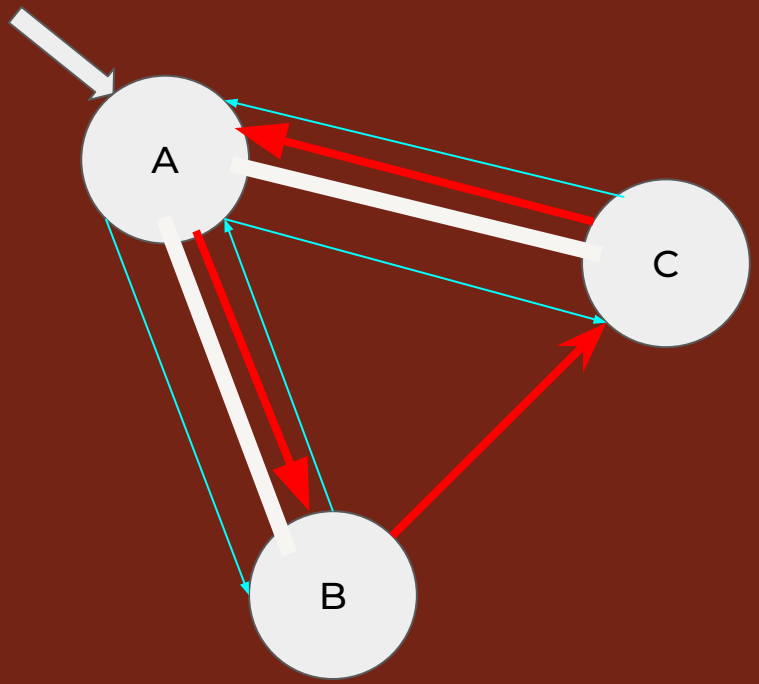
Princípio base: Ir diretamente de A a B é sempre pelo menos tão rápido como ir de A a B passando por C.

Procedimento:

- Gerar Minimum Spanning Tree (MST)
- Percorrer a MST em depth-first search, armazenando a sequência de vértices.

Desafio: gerar a MST não é trivial devido à maneira como o grafo está estruturado. Experimentaram-se os algoritmos de Kruskal e Prim, tendo-se decidido que o segundo seria mais prático.

Exemplo:



Complexidade: O(E * log(V))
Depende do algoritmo para gerar a MST, pois a DFS é O(V + E)

Legenda:
A branco: arestas da MST
A vermelho: arestas do ciclo final
A azul: "percurso" da DFS

Algoritmos Grafos Completamente Ligados

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Algoritmos para grafos completamente ligados

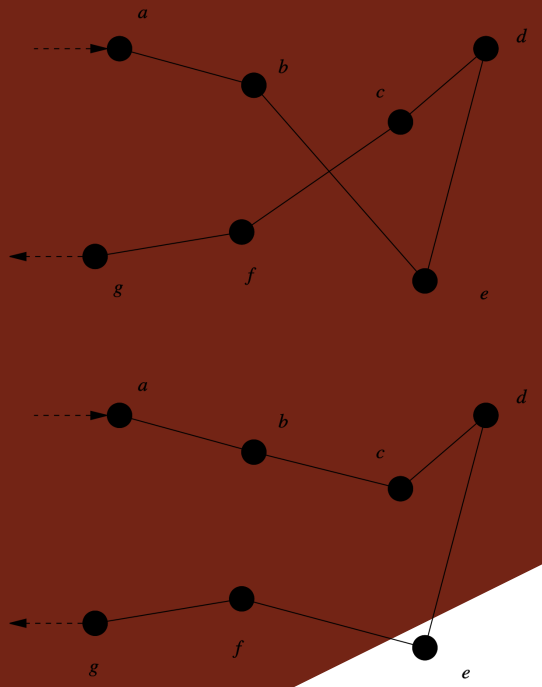
- **“HUB”:**
  - A qualquer momento, ter um ciclo, e ir acrescentando vértices na posição que a um dado momento minimize o seu comprimento.
- **“Dijkstra”:**
  - Determinar o caminho mais curto para cada vértice. Ligar um vértice final da árvore ao inicial, e inserir em sequência cada ramo de forma a formar um ciclo.
- **“DFS”:**
  - Começando num dado vértice, continuar uma pesquisa em profundidade tal que a cada momento seja escolhida a edge menos comprida para um vértice não visitado.

Características e problemas encontrados:

- **“HUB”:**
  - Não obtém os comprimentos mais curtos possíveis, sendo estes contudo melhores que os obtidos pelo método que utiliza o algoritmo de Dijkstra.
  - Em grafos não completamente ligados, quanto menos arestas o grafo tem, mais provável é este falhar, produzindo o comprimento de um ciclo incompleto..
- **“Dijkstra”:**
  - Quantas mais arestas tem um grafo, mais provável é de resultar num resultado não ótimo - não é bom para grafos completamente ligados.
- **“DFS”:**
  - Rápido em termos de performance, mas é apenas confiável em grafos completamente ligados, pois não é capaz de gerar um resultado, mesmo aproximado, para outros tipos de grafos.

2-opt - O(V²)

- Um algoritmo de otimização para o problema de TSP.
- Consiste em percorrer os vértices de uma possível solução para o problema e, para cada um, analisar todos os outros vértices para verificar se caso estes 2 vértices fossem trocados de ordem na solução se é possível chegar a um resultado melhor.
- No exemplo -> ((b-c) + (f-e)) < ((b-e) + (f-c)), logo trocando a ordem desde e até c no path da solução obtém-se uma solução melhor.



U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Grafos
“Real World”

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Algoritmo Greedy adaptado $O(V^2)$

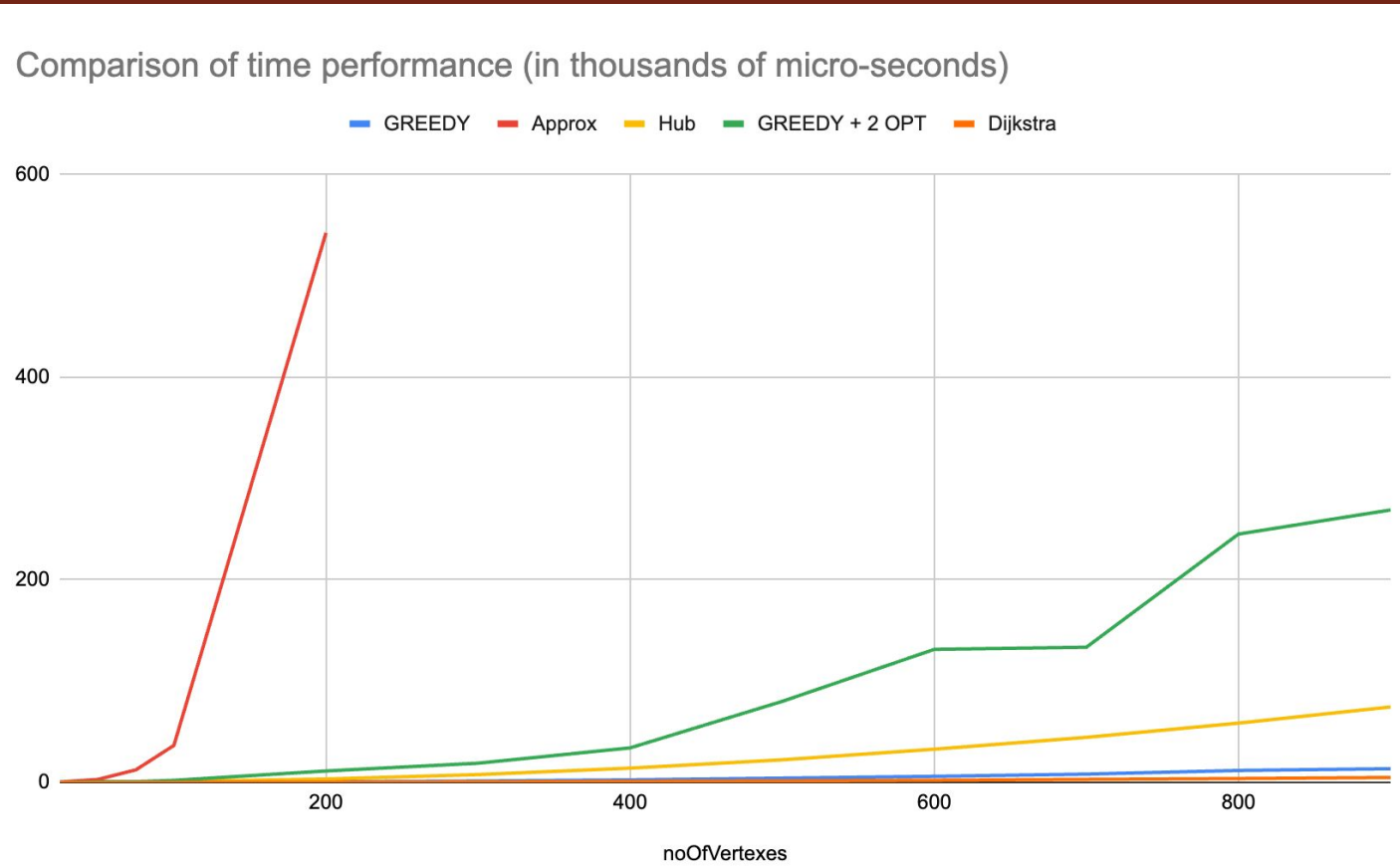
Neste algoritmo, em cada Vertex, encontra-se o Vertex mais próximo que ainda não tenha sido visitado.

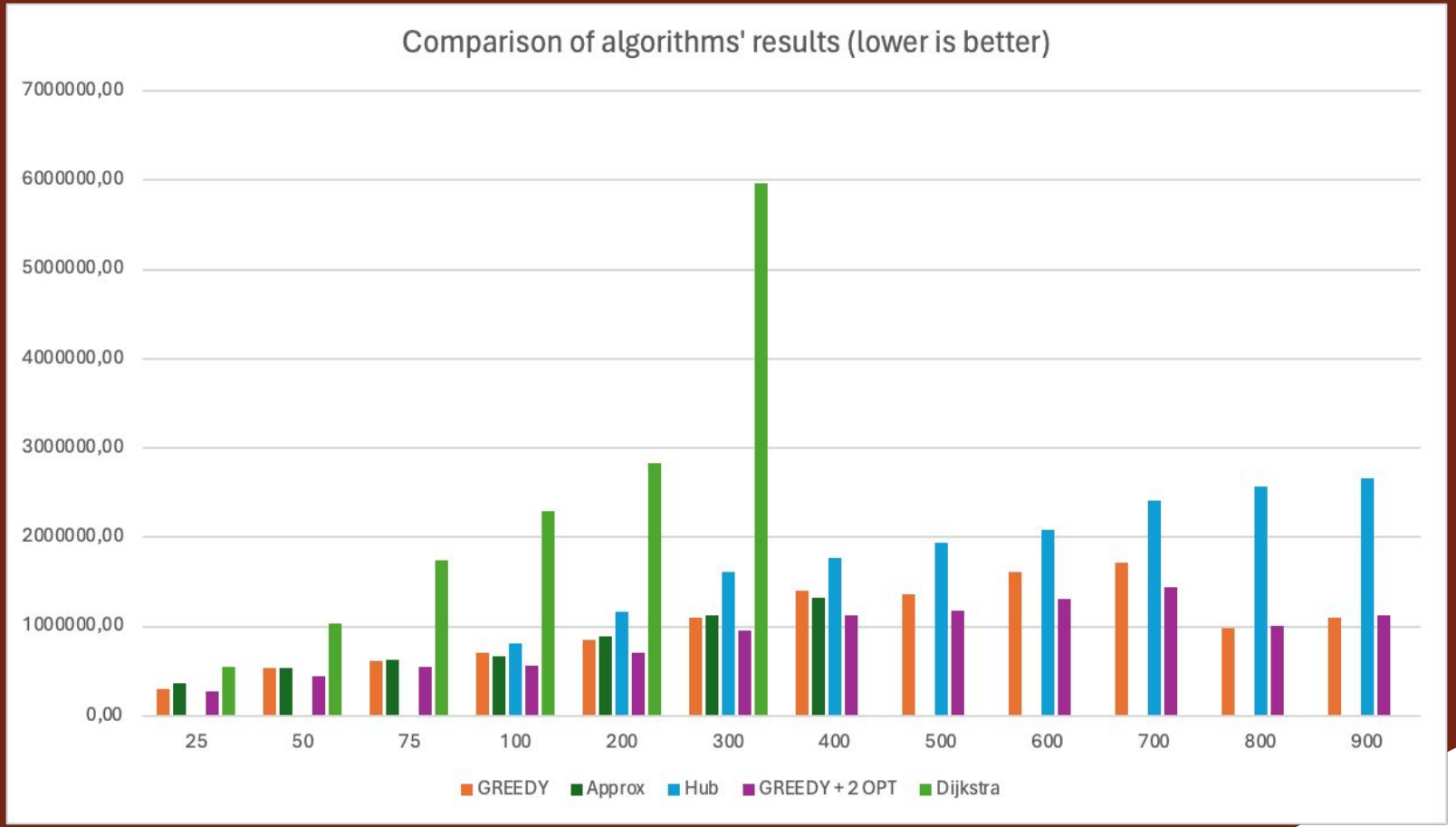
Caso a certa altura não seja possível visitar mais nenhum Vertex, sem ter de re-visitar um Vertex já visitado, possibilitamos que o utilizador escolha duas opções:

- Retornar para a origem
- Visitar o seguinte Vertex através da ida para outro Vertex auxiliar.

Estatísticas

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO


Comparison of time performance (in thousands of micro-seconds)
GREEDY
Approx
Hub
GREEDY + 2 OPT
Dijkstra
600
400
200
0
200
400
600
800
noOfVertexes


Comparison of algorithms' results (lower is better)
7000000,00
6000000,00
5000000,00
4000000,00
3000000,00
2000000,00
1000000,00
0,00
25
50
75
100
200
300
400
500
600
700
800
900
GREEDY
Approx
Hub
GREEDY + 2 OPT
Dijkstra

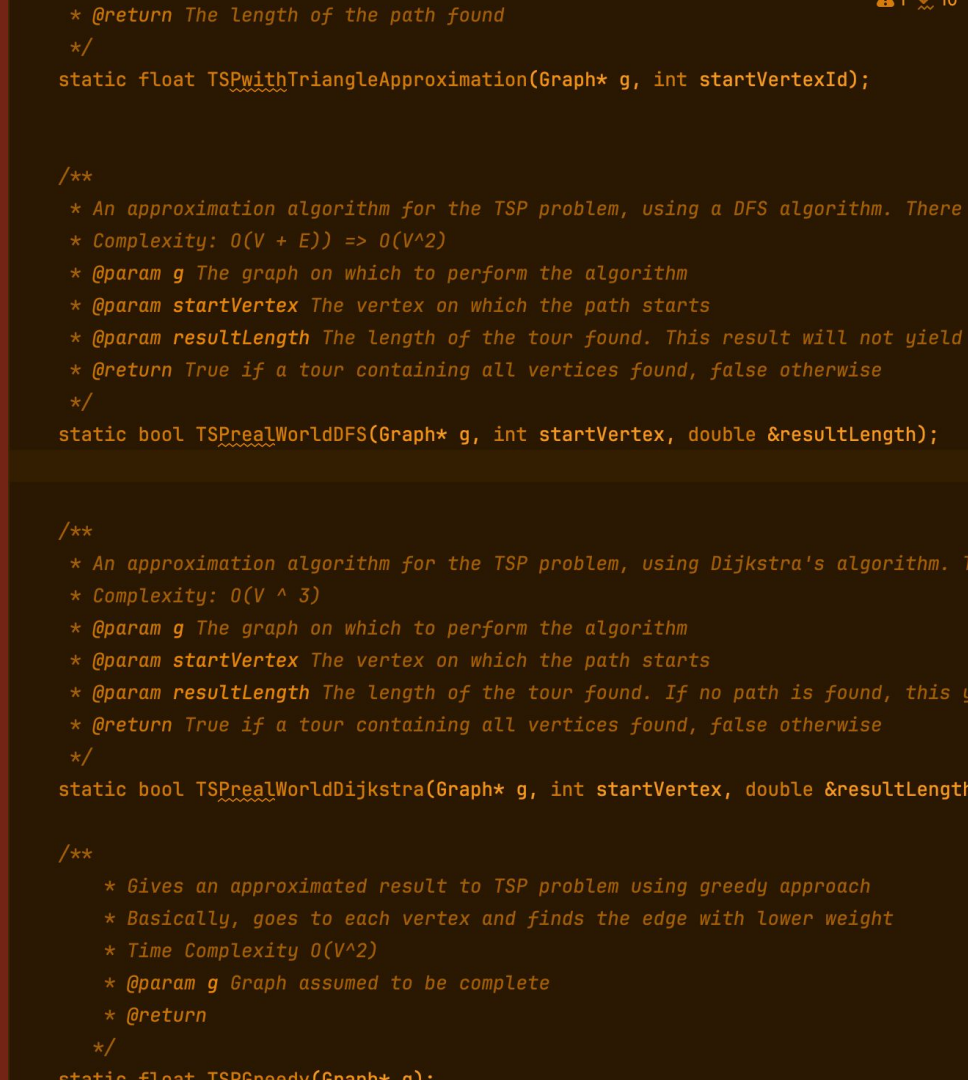
O que ainda tentámos?

Christofides Algorithm

- Tentámos implementar ainda o algoritmo da heurística de Christofides por acreditarmos que poderia dar um bom resultado neste tipo de grafos baseado em distâncias reais.
- No entanto, um dos passos finais deste algoritmo passa por calcular um ciclo euleriano, o que se revelou um problema difícil para encontrar uma boa solução em feasible time e, por isso, resolvemos optar por algoritmos alternativos já apresentados.

Documentação

Doxygen



Reflexões:

- Em grafos não completamente ligados, encontrar ciclos contendo todos os vértices provou-se complicado, tendo sido feitas aproximações com tendência a resultar em “falsos fracassos”.
- O tamanho dos grafos mais complexos obrigou a uma pesquisa sobre diferentes heurísticas e algoritmos visando reduzir a complexidade, e forneceu uma melhor compreensão da dificuldade de encontrar soluções para problemas em escala real.

Participação

Acreditamos que todos os membros trabalharam de uma forma equilibrada.

Fontes:

Solving Traveling Salesman Problem With a Non-complete Graph, Mahsa Sadat Emami Taba, University of Waterloo

Travelling salesman problem, Wikipedia