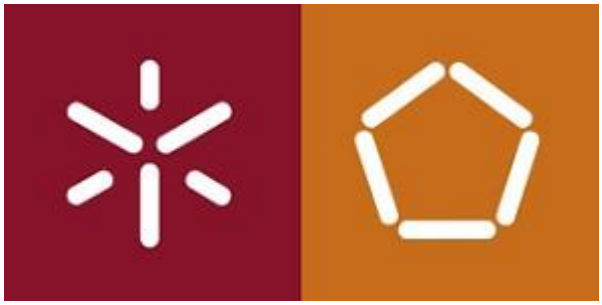
UNIVERSIDADE DO MINHO



INTELIGÊNCIA ARTIFICIAL

TRABALHO PRÁTICO

2023/2024

Autores:

A100646, Diogo Abreu

A100549, Luís Figueiredo

A100835, Miguel Gramoso

A100706, Rodrigo Monteiro

ÍNDICE

Tabela de Figuras

1. Introdução

Este trabalho foi realizado no âmbito da unidade curricular de Inteligência Artificial da Licenciatura em Engenharia Informática da Universidade do Minho, e teve como objetivo a resolução de problemas através da conceção e implementação de algoritmos de procura. Neste trabalho foram implementadas estratégias para a resolução de problemas com o uso de algoritmos de procura informada e não informada, partindo da formulação do problema em questão. Ao longo deste documento vai ser explicado todo o processo de desenvolvimento dos mecanismos de procura implementadas neste projeto.

2. Formulação do Problema

Uma vez que o problema se encontra num ambiente determinístico e completamente observável, em que o agente "sabe" exatamente o estado em que estará e as soluções pretendidas são sequências, podemos afirmar que este é um problema de estado único. Além disso, a sua formulação pode ser efetuada da seguinte forma:

- Representação do estado: Grafo não orientado, em que cada nodo representa uma interseção entre uma ou mais ruas

- Estado inicial: A posição inicial representa o local de início de onde o estafeta parte

- Estado/teste objetivo: Alcançar um ponto de entrega associada à morada atribuída à encomenda a ser entregue pelo estafeta

- Operadores: Deslocação entre nodos adjacentes

- Solução: Um caminho válido (sequência de posições percorridas) que comece na posição inicial do estafeta (inicialmente a posição gerada automaticamente pelo programa e que depois tose sempre o último ponto de entrega da encomenda anterior) e termine num ponto de entrega da morada associada à encomenda

- Custo da solução: Distância total do percurso feito pelo estafeta

3. Criação do Mapa

O mapa é criado através de um método denominado *inicializa_mapa* onde neste se cria uma lista de objetos da classe Rua, associada a um objeto da classe freguesia. Ambas as classes surgem no ficheiro *Cidade.py*.

Cada rua tem um nome, dois pontos de coordenadas (x1,y1) e (x2,y2) que servirão tanto de limite da rua como de ponto de entrega na rua, uma velocidade máxima na rua (será explicada no final do relatório) e uma freguesia.

```
Rua( nome: "Avenida da Liberdade", y1: 0, x1: 0, y2: 0, x2: 200, velocidade_maxima: 80, freguesia: "Gualtar")
```

Figura 1: *Exemplo de uma instância de objeto Rua*

Cada freguesia tem um nome e dois pontos de coordenadas (x1,y1) e (x2,y2). No mapa, cada uma destas será visível através do formato quadrangular.

```
Freguesia( nome: "Gualtar", y1: 0, x1: 0, y2: 200, x2: 200)
```

Com a lista de ruas criada e o recurso à classe Canvas (Tkinter) a função *desenha_mapa* do ficheiro *Cidade.py* faz uma representação gráfica do mapa de acordo com os dois pontos de coordenadas em cada instância de objeto Rua presente na lista de ruas guardada na classe *Mapa* na forma de uma linha. Cada linha guarda uma tag única no formato string para posteriormente ser possível realizar o desenho do caminho de nodos solucionado pelos algoritmos utilizados no programa.

Desta forma, as ruas serão representadas por linhas, podendo estender-se para além de uma interseção ou não. Estas poderão ser horizontais, verticais ou mesmo diagonais.

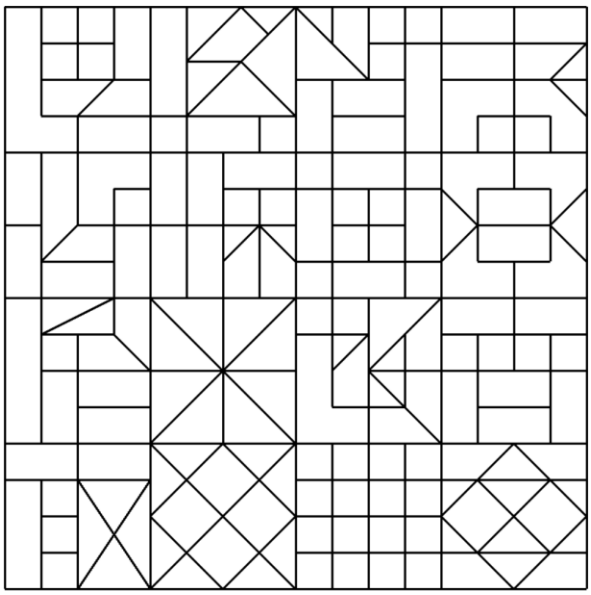


Figura 2*: Mapa da Cidade*

O mapa tem assim 16 freguesias e mais de uma centena de ruas.

Cada rua tem associada dois pontos de entrega, uma em cada ponta da rua

4. Criação do Grafo

O grafo é criado a partir do método *inicializa_grafo* presente na classe Grafo. Este método introduz num dicionário todos os nodos e respetivas ligações a esse nodo que serão utilizados para calcular o caminho do estafeta até a um ponto de entrega da morada associada à encomenda.

```
self.nodos["100-0"] = ["50-0", "100-50", "150-0"]
```

Figura 3: *Exemplo de ligações possíveis a um nodo*

Deste modo, cada nodo é um ponto de coordenadas no mapa que faz a ligação entre uma ou mais ruas. É representado no seguinte formato: "x-y" sendo x a abcissa do ponto de coordenadas e y a ordenada do mesmo.

Cada aresta do grafo tem um custo igual à distância entre as coordenadas dos dois pontos associados a esses dois nodos que fazem a aresta.

5. Funcionamento do Programa

5.1 Menus

O programa começa por apresentar ao administrador o seguinte menu:

Figura 4: *Menu Inicial do Programa*

O menu apresenta duas opções:

1. Gestão App – Onde é depois possível visualizar toda a informação referente às encomendas entregues e aos estafetas.

2. Sair – Botão para sair do programa.

Prosseguindo através da opção 1, o administrador depara-se com outro menu:

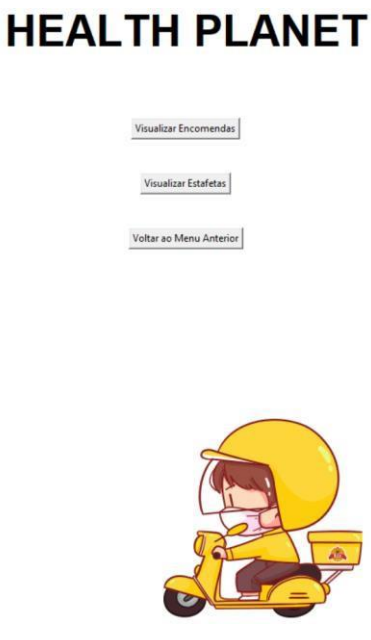


Figura 5: *Menu da App*

Temos 3 opções:

1. Visualizar Encomendas – Onde é possível depois ter acesso às informações referentes a cada tabela.

2. Visualizar Estafetas – Onde é possível depois ter acesso às estatísticas de cada estafeta.

3. Voltar ao Menu Anterior – Volta ao Menu Inicial.

Escolhendo a opção 1, temos acesso a uma tabela com as 1000 encomendas entregues ao longo do programa.

Figura 6: *Tabela de Encomendas Entregues*

Em cada linha da tabela temos informação relativa a uma encomenda com o seu ID, Peso, Morada e Estafeta. O administrador ao clicar numa linha da tabela terá depois acesso ao caminho feito pelo estafeta para entregar a encomenda (inicialmente solucionado com recurso ao algoritmo A*).

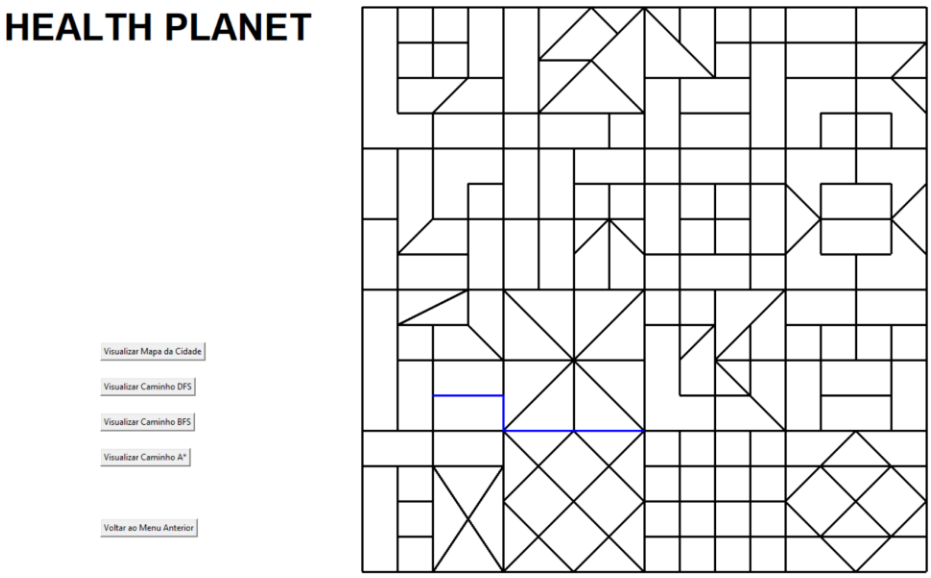


Figura 7*: Caminhos associados a um algoritmo*

É possível agora observar que temos 4 botões (excluindo o de voltar ao menu anterior). O botão *visualizar mapa* dá acesso ao mapa com uma *combobox* com todas as moradas existentes no programa. Clicando numa delas, a rua fica *highlighted* a vermelho no mapa como na seguinte imagem:

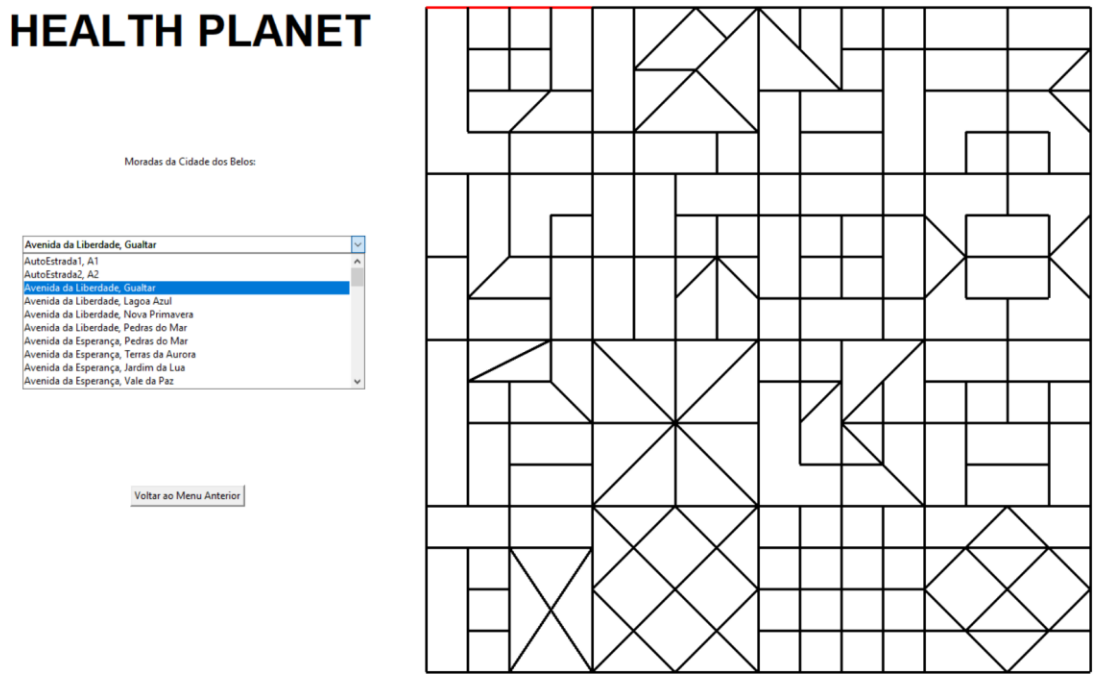


Figura 8: *Mapa e moradas highlighted*

Os outros 3 botões funcionarão tal como na figura anterior a esta, em que o caminho *highlighted* a azul será uma solução do algoritmo DFS, BFS ou A*.

Relativamente à opção de *Visualizar Estafetas* temos acesso a uma tabela com as estatísticas dos estafetas no final de todas as entregas. É possível ordenar a tabela por *rating* ou *distância* clicando no cabeçalho de cada coluna respetiva.

6. Estratégias de Procura Não Informada

6.1 Procura em Profundidade (DFS)

Este tipo de procura usa como estratégia, expandir sempre um dos nodos mais profundos do grafo. As **vantagens** deste tipo de procura são:

- Pouco uso de memória.
- Bom para problemas com múltiplas soluções, pois a probabilidade de estar a procurar por um caminho possível aumenta.

As **desvantagens** deste tipo de procura são:

- Pouca eficiência em grafos com uma profundidade elevada e poucas soluções.
- A solução obtida pode não ser a solução ótima.

No entanto, no nosso problema o estafeta terá que obter um caminho ideal de forma rápida para evitar perder tempo entre cada entrega, logo esta estratégia muito provavelmente não irá conseguir obter um percurso ótimo de forma relativamente eficiente. A nossa implementação do algoritmo de procura em profundidade impede que os nodos que já tenham sido visitados sejam explorados pelo algoritmo de maneira a evitar a ocorrência de loops no processo de pesquisa. Esta estratégia de procura apresenta as seguintes propriedades:

- Tempo: $O(b^{\mathrm{m}})$
- Espaço: $O(bm)$

Onde **b** é o número máximo de sucessores de um nodo da árvore de procura e **m** a máxima profundidade do espaço de estados.

6.1.1 Exemplo de uma Procura em Profundidade (DFS)

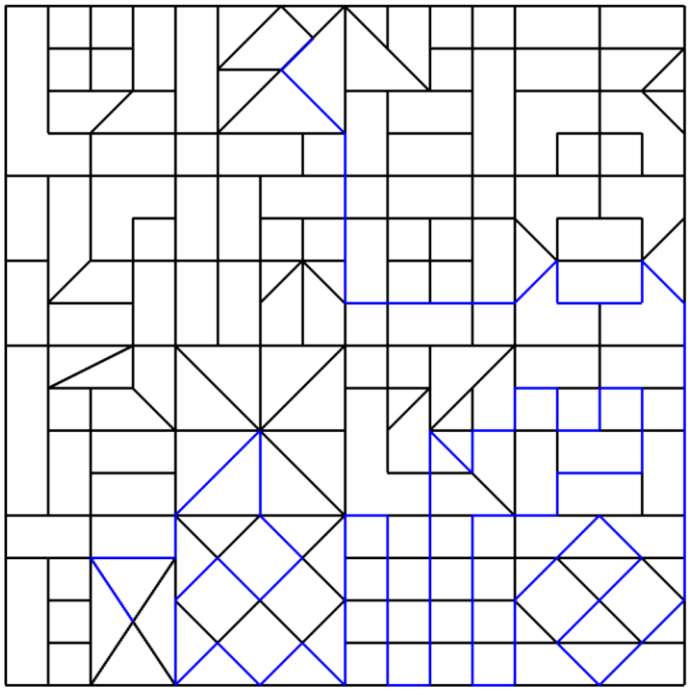


Figura 9: *Exemplo de uma Procura em Profundidade*

Como podemos ver na figura acima, o algoritmo de procura em profundidade obteve um caminho não ideal devido à sua estratégia de expandir sempre para nodos mais profundos do grafo, não sendo a melhor opção para este tipo de problemas uma vez que existem ramificações bastante longas que não vão ao encontro de uma solução ótima.

6.2 Procura em Largura (BFS)

Este tipo de procura expande primeiro os nodos de menor profundidade do grafo. A **vantagem** deste tipo de procura são:

- Solução ótima quando todas as arestas têm custo 1.

As **desvantagens** deste tipo de procura são:

- Tempo de pesquisa elevado visto que tende a percorrer muitos mais nodos do que os necessários para criar o caminho válido.
- Ocupa muito espaço em memória.

No nosso problema o tamanho do grafo é bastante grande, o que faz com que este algoritmo também não seja o melhor para o programa. Esta estratégia de procura apresenta as seguintes propriedades:

- Tempo: $O(b^d)$
- Espaço: $O(b^d)$

Onde **b** é o número máximo de sucessores de um nodo da árvore de procura e **d** a profundidade da melhor solução.

6.2.1 Exemplo de uma Procura em Largura (BFS)

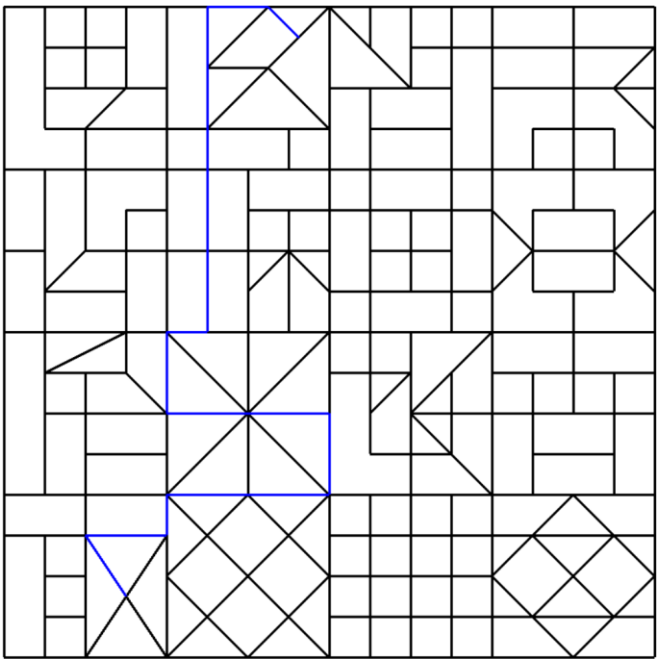


Figura 10: *Exemplo de uma Procura em Largura*

Como podemos ver na figura acima, o algoritmo de procura em largura também não obteve um caminho ótimo uma vez que no nosso problema nem todas as arestas têm o mesmo custo, o que faz com que este algoritmo não seja o ideal para achar soluções ótimas no contexto do nosso problema.

7. Estratégias de Procura Informada

No nosso programa implementámos a **Procura A*** (como *default*) e a **Procura Greedy**.

O nosso objetivo era implementar uma heurística baseada na distância do nodo atual até ao nodo final e na velocidade possível em cada rua. No entanto, esta última parte não foi implementada no programa. Ficámos então apenas pela heurística baseada na menor distância em linha reta do nodo atual ao nodo final.

7.1 Procura Greedy

A Procura Greedy, ou busca gulosa, é um algoritmo de procura informada que escolhe o próximo nodo a ser explorado com base apenas na heurística, sem considerar o custo acumulado até o momento. A função de escolha é definida como f(n)=h(n) onde h(n) é a heurística que estima o custo para alcançar o objetivo a partir do nodo n.
Vantagens da Procura Greedy:

- Eficiente em termos de tempo, já que apenas considera a heurística.
- Requer menos espaço em memória em comparação com A*.

Desvantagens da Procura Greedy:

- Não garante a solução ótima, mesmo se a heurística for admissível.
- Pode ficar presa em mínimos locais.

Propriedades da Procura Greedy:

- Tempo: $O(b^m)$ (com uma boa função heurística pode diminuir significativamente)
- Espaço: $O(b^m)$ (mantém todos os nós em memória)

Em resumo, a Procura Greedy é rápida, mas pode sacrificar a otimalidade em favor da eficiência.

7.1.1 Exemplo de uma Procura Greedy

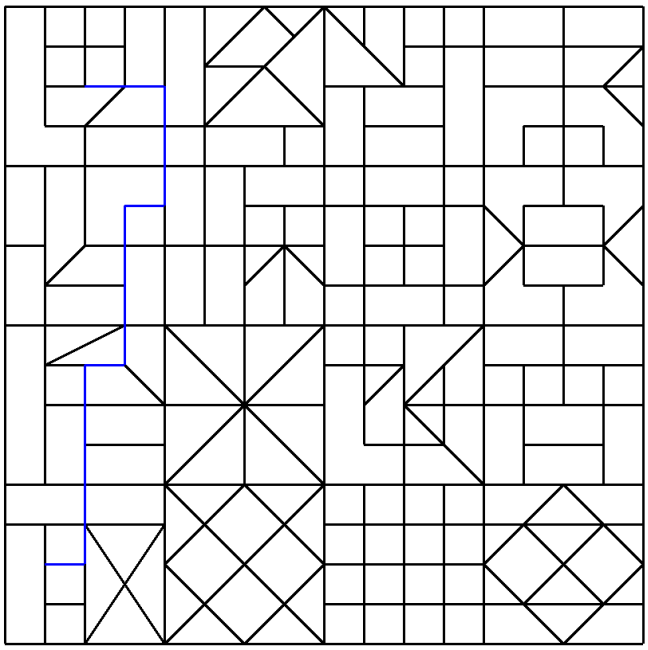


Figura 11: *Exemplo de uma Procura Greedy*

7.2 Procura A*

Este tipo de procura evita expandir caminhos que são dispendiosos, combinando para isso o algoritmo de procura gulosa com o algoritmo de procura uniforme. Utiliza então a seguinte função para a escolha do nodo a ser explorado de seguida:

$f(n) = g(n)+h(n)$

Onde:

- g(n) = custo acumulado
- h(n) = custo estimado para chegar ao destino (heurística)

As **vantagens** deste tipo de procura são:

- Obtém a solução ótima se a sua heurística for admissível.

As **desvantagens** deste tipo de procura são:

- Ocupa muito espaço em memória.

Este algoritmo de procura para a escolha do próximo nodo, analisa a heurística e o custo acumulado do seu percurso até ao momento. Isso permite ao algoritmo encontrar a solução ótima para o problema.

Esta estratégia de procura apresenta as seguintes propriedades:

- Tempo: exponencial em (erro relativo de h * comprimento da solução)
- Espaço: $O(b^m)$ (mantém todos os nós em memória)

Onde b é o número máximo de sucessores de um nodo da árvore de procura e m a máxima

profundidade do espaço de estados.

7.2.1 Exemplo de uma Procura A*

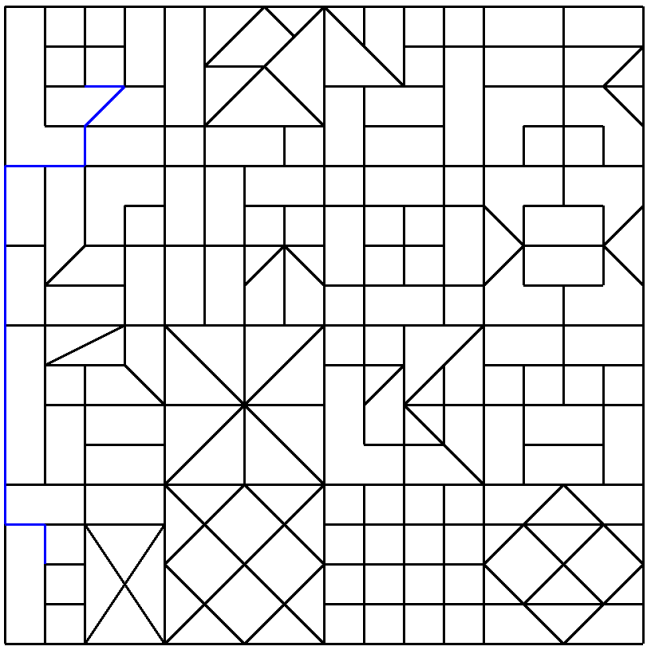


Figura 12: *Exemplo de uma Procura A**

8. Conclusão

Concluindo, neste projeto o nosso grupo foi capaz de implementar algoritmos de procura de forma a explorar caminhos possíveis para a entrega de cada encomenda. Foi possível também melhorar a nossa capacidade quanto à implementação de interfaces gráficas que dão um aspeto ebeleza melhores ao programa.

O nosso projeto infelizmente não conseguiu realizar um objetivo que melhoraria o trabalho que era a parte da influência da velocidade no projeto. Era da nossa vontade implementar esta parte na heurística e também em termos de trânsito (em que o programa daria valores arbitrários de velocidade média em cada rua). Tal é o facto que implementámos estradas cujos valores máximos de velocidade variam.

No entanto, pensamos que conseguimos desenvolver um bom projeto satisfazendo as condições mínimas previstas inicialmente.