Inteligência Artificial

2016/2017 - 3º Ano, 2º Semestre

Pesquisa aplicada à recolha de lixo numa cidade

Relatório final

Turma 5, Grupo A3_1

Hélder Antunes, up201406163 - up201406163@fe.up.pt

Miguel Lucas, ei11140 - ei11140@fe.up.pt

Renato Abreu, up201403377 - up201403377@fe.up.pt

21 de Maio de 2017

Índice

1. Objetivo

No âmbito da cadeira Inteligência Artificial do Mestrado Integrado de Engenharia Informática e Computação, propusemo-nos a desenvolver um projeto utilizando pesquisa informada de soluções aplicada à recolha de resíduos numa cidade.

O objetivo final do trabalho será desenvolver um programa que calcule as melhores rotas para os camiões, desde a central de camiões até à estação de tratamento. Entende-se por melhores rotas aquelas que permitem maximizar a recolha de lixo, e, ao mesmo tempo, minimizar a distância percorrida.

Para atingirmos esse objetivo, serão aplicados diversos algoritmos de pesquisa, entre os quais a pesquisa em profundidade e largura (métodos fracos) e o algoritmo A* (método informado). Será depois comparada a qualidade de soluções encontrada pelos diversos algoritmos.

2. Especificação

2.1. Análise detalhada do problema

O problema terá como input alguns dados relativos à situação em causa:

- mapa da região, representado em forma de grafo;
- posição da central de camiões;
- posição da estação de tratamento do lixo;
- posições dos contentores do lixo e a quantidade lixo que contêm;
- capacidade dos camiões disponíveis (cada camião transporta apenas um tipo de lixo: papel, plástico, vidro e lixo comum)

Analisando esses dados, o programa terá que, para cada determinado tipo de lixo (papel, plástico, vidro ou lixo comum), determinar as rotas de recolha a realizar pelos camiões desde a central de camiões até à estação de tratamento do lixo.

A solução encontrada deve considerar apenas os contentores com resíduo suficiente que justifique a sua recolha, de forma a minimizar a distância percorrida e maximizar o lixo recolhido.

2.1.1 Cenários do problema

A aplicação terá que saber fazer uso de vários camiões. Podem existir situações em que usar vários camiões seja menos custoso, enquanto que noutras um camião é suficiente.

Por exemplo, existem várias agregações de contentores A_1, A_2, ..., A_N, e a distância entre eles é de tal ordem elevada que torna impraticável um camião viajar entre duas agregações. Nesta situação, a solução mais vantajosa seria utilizar N camiões, em que cada camião C_i faz o percurso da central de camiões até à agregação A_i, visitando todos os contentores dessa agregação, e daí vai até à estação de tratamento do lixo.

Outro cenário seria a baixa quantidade de resíduos num contentor de modo que não compensa visitar e consequente recolher o lixo desse contentor.

2.2. Abordagem

Usando o algoritmo A* para solucionar o problema, considerou-se a seguinte representação de estados:

$$\{nó\ no\ grafo,\ distância\ percorrida,\ lixo\ recolhido\}$$

O **estado inicial** é: $\{nó\ da\ central\ de\ camiões,\ 0,\ 0\}$

O **estado final** é : $\{nó\ da\ estação\ de\ tratamento\ do\ lixo,\ D,\ L\}$

Podem existir vários estados finais, em que D, L, ou ambos diferem entre esses estados.

As **funções de transição** dizem respeito às ações que o camião pode efetuar. Em cada estado, um camião pode:

- recolher lixo (se no local existir lixo que possa ser carregado pelo camião) aumentando o lixo recolhido e mover-se para um nó adjacente.
- mover-se apenas para um nó adjacente.

Função heurística: $f'(n) = g(n) + h'(n)$

$$g(n) = a * distância\ percorrida\ /\ soma\ das\ arestas - b * lixo\ recolhido\ /\ capacidade\ de\ um\ camião$$

As variáveis distância percorrida e lixo recolhido têm unidades de medida diferentes. Para retirar as unidades de medida às variáveis, dividiu-se ambas pelo seu valor máximo possível.

$$h'(n) = a * distância\ à\ estação\ de\ tratamento\ /\ soma\ das\ arestas - b * proximidade\ de\ lixo\ /\ lixo\ total$$

A $proximidade\ de\ lixo$ calcula-se da seguinte maneira:

1. Constrói-se um grafo reverso G_r do grafo original G.
2. Para cada nó N_i pertencente a G_r com um contentor com lixo corre-se um algoritmo de pesquisa em largura que coloca N_i com fator de proximidade igual ao lixo do contentor e os restantes nós com fator de proximidade igual a $proximidade\ de\ lixo\ do\ nó\ pai\ /\ 2$.
3. No fim coloca-se os fatores de proximidade de lixo dos nós de G iguais aos nós de G_r.

Os valores a e b são valores manipuláveis que permitem priorizar a distância ou o lixo recolhido, tendo a restrição: $a + b = 1$.

Alguns valores previamente apresentados são pré-calculados no início do algoritmo. Entre esses valores estão: soma das arestas, armazenamento de um camião, lixo total e distância à estação de tratamento de lixo.

A distância até à estação de tratamento de cada nó calcula-se correndo o algoritmo de Dijkstra no grafo reverso do grafo original com origem no nó da estação de tratamento de lixo.

O algoritmo A* procura um caminho apenas para um camião. Numa solução com vários caminhos possíveis, verifica-se se existe contentores com mais de metade do seu espaço ocupado. Se sim, corre-se de novo o algoritmo para outro camião. Caso contrário, termina-se o processamento.

3. Desenvolvimento

A linguagem de programação escolhida foi Java. Utilizou-se GraphStream, uma biblioteca para desenho de grafos (http://graphstream-project.org/).

3.1. Arquitetura da aplicação

A aplicação contém os seguintes packages Java:

- wasteManagement - parser de ficheiros e armazenamento de informação
- myGraph - implementação de grafos e algoritmos que operam sobre ele (A*, dfs, bfs, Dijkstra)
- gui - interface gráfica para o utilizador (interface Swing)

3.2. Detalhes relevantes da implementação

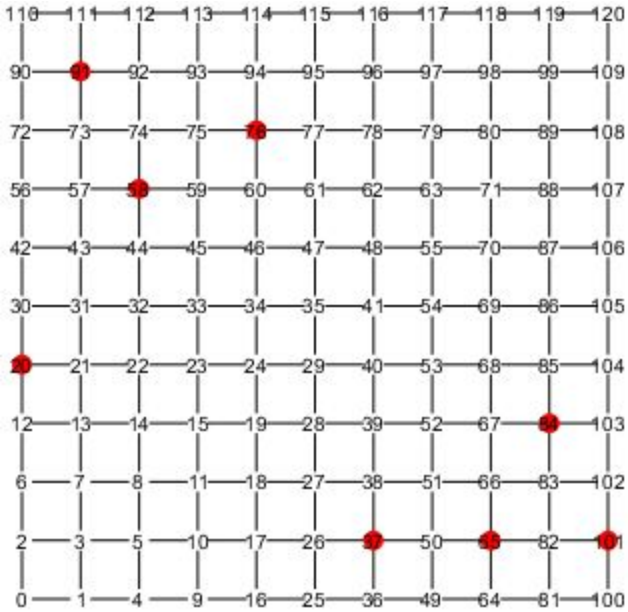
A aplicação recebe como input dois tipos de ficheiros. Um tipo é do formato '.dgs' e representa um grafo (mapa da região), e o segundo tipo é do formato '.xml' e contém informação sobre os resíduos existentes e respetivas posições dos contentores, capacidade dos camiões, e as posições do nó da central de camiões e do nó da estação de tratamento do lixo. Esses ficheiros de input encontram-se na pasta /data.

Na implementação dos algoritmos, foi atribuído um número de tentativas de chegar à solução igual a 3. Isto significa que quando a solução chega a um estado final (o nó representa a estação de tratamento de lixo), esse estado é guardado, e continua-se o processamento. No fim compara-se as soluções obtidas e escolhe-se aquela em que o rácio *lixo recolhido / distância percorrida* é maior.

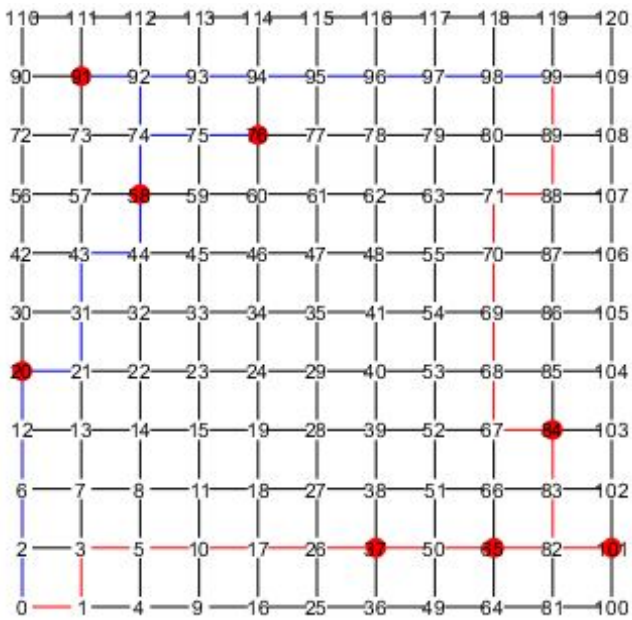
4. Experiências

1ª experiência

A primeira experiência consistiu em testar o programa na forma como dispõe camiões por vários percursos. Na figura seguinte os nós vermelhos contêm resíduos, a central de camiões situa-se no nó 0 e a estação de tratamento de lixo no nó 99.



Grafo original



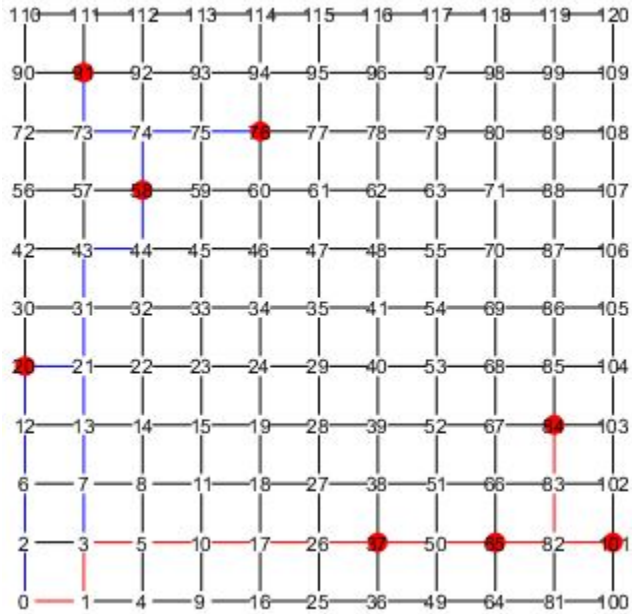
Caminhos gerados pelo A*

Resultados dos diferentes algoritmos para a 1ª experiência			
Algoritmo	Lixo recolhido (Kg)	Distância percorrida (Km)	Lixo recolhido / Distância percorrida (Kg / Km)
A*	80.00	46.00	1.74
DFS	80.00	242.00	0.33
BFS	80.00	130.00	0.62

2ª experiência

A segunda experiência consistiu em colocar a estação de tratamento do lixo próximo da central de camiões, e verificar se os camiões conseguiam ir aos contentores mais afastados, ou seja, não iam diretamente para a estação de tratamento do lixo sem recolher lixo.

O grafo apresentado na figura seguinte mostras os caminhos obtidos. O nó 0 é a central de camiões e o nó 3 é a estação de tratamento do lixo.



Caminhos gerados pelo A*

A solução encontrada demonstra que os percursos efetuados afastem-se o suficiente para procurar lixo.

Resultados dos diferentes algoritmos para a 2ª experiência			
Algoritmo	Lixo recolhido (Kg)	Distância percorrida (Km)	Lixo recolhido / Distância percorrida (Kg / Km)
A*	80.00	50.00	1.60
DFS	80.00	550.0	0.15
BFS	80.00	86.0	0.93

5. Conclusões

Tal como esperado, o algoritmo A* com as heurísticas já mencionadas consegue obter melhores resultados que os algoritmos DFS e BFS.

Por outro lado, o algoritmo BFS conseguiu nas experiências anteriores obter melhores resultados que o DFS, principalmente na distância percorrida. Isto deve-se ao facto que o algoritmo DFS demorar muito a chegar a um estado final (por exemplo se percorrer as arestas todas antes de ir para o nó da estação de tratamento de lixo), e por isso, obter distância percorridas muito grandes. Pelo contrário, o BFS alcança um estado final pelo caminho mínimo, se o peso das arestas for constante (nas experiências apresentadas o peso das arestas era constante).

Podemos concluir que o uso de métodos informados para encontrar soluções é muito vantajoso em situações nas quais a própria solução depende de várias variáveis. Contudo, a definição de uma boa heurística que tenha em conta todos os detalhes do problema é essencial para encontrar a melhor solução possível.

6. Melhoramentos

Um possível melhoramento seria adicionar a variável combustível ao estado de uma solução e adicionar postos de combustível no grafo. Assim, seria necessário modificar as heurísticas do algoritmo A* de modo a que também se tivesse em conta o combustível atual. Por outro lado, teria de se abastecer apenas em condições vantajosas (a distância até a um posto de abastecimento é curta e a quantidade de combustível restante é baixa).

Em suma consideramos que a introdução de uma nova variável e a consequente alteração das heurísticas seria um desafio interessante.

7. Recursos

7.1. Bibliografia

- Slides das aulas teóricas.
- Moore, Andrew. A-star Heuristic Search. Disponível em: http://www.cs.cmu.edu/~./awm/tutorials/astar.html [Acessado no dia 26/3/2017]
- Patel, Amit. Introduction to A*. Disponível em: http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html [Acessado no dia 26/3/2017]

7.2. Software

- GraphStream - A Dynamic Graph Library [http://graphstream-project.org/]
- IntelliJ IDEA - [https://www.jetbrains.com/idea/]

7.3. Contribuição e distribuição do trabalho

Hélder Antunes, up201406163 - 40% (implementação do grafo, heurística, realização de experiências e relatório)

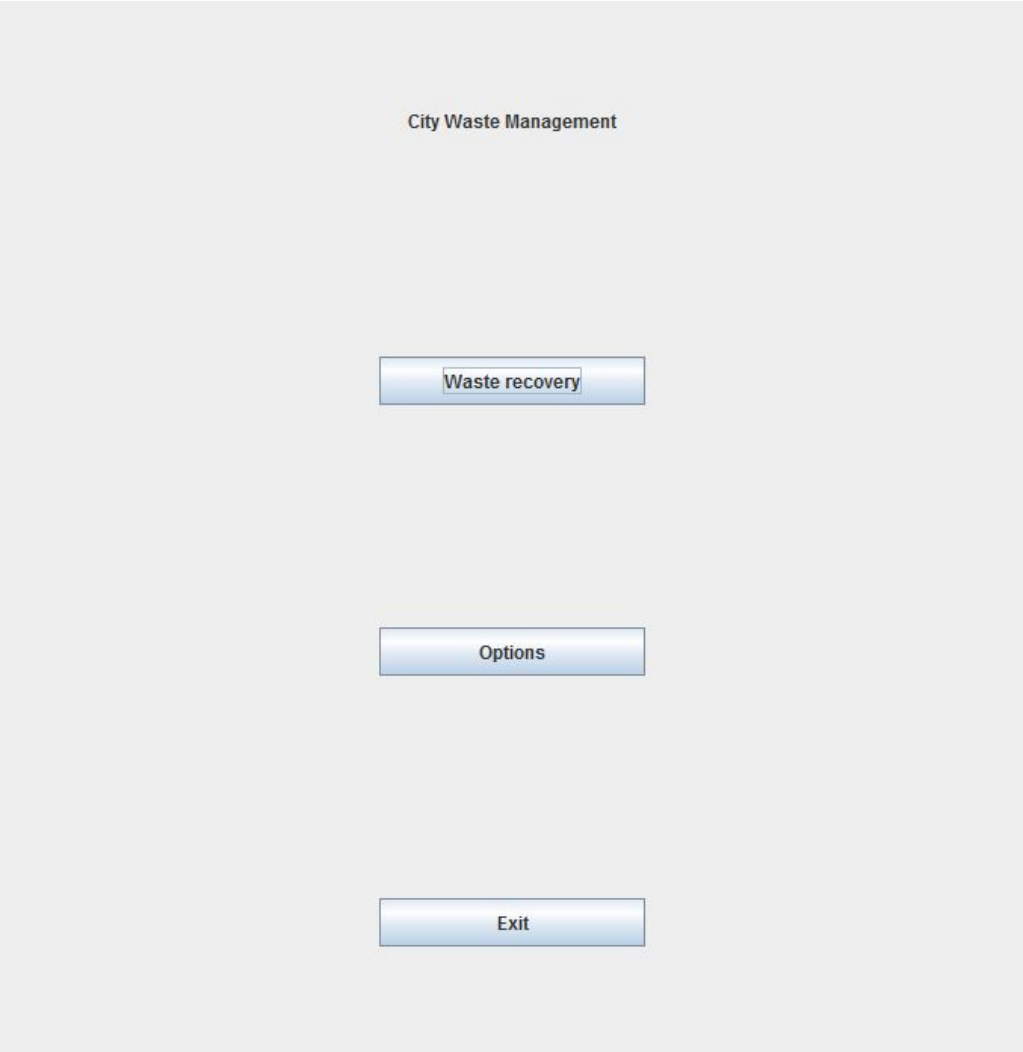
Miguel Lucas, ei11140 - 20% (heurística, realização de experiências e relatório)

Renato Abreu, up201403377 - 40% (realização de experiências, heurística, parte gráfica e relatório)

8. Apêndice

8.1. Manual do utilizador

A aplicação começa com um menu principal:

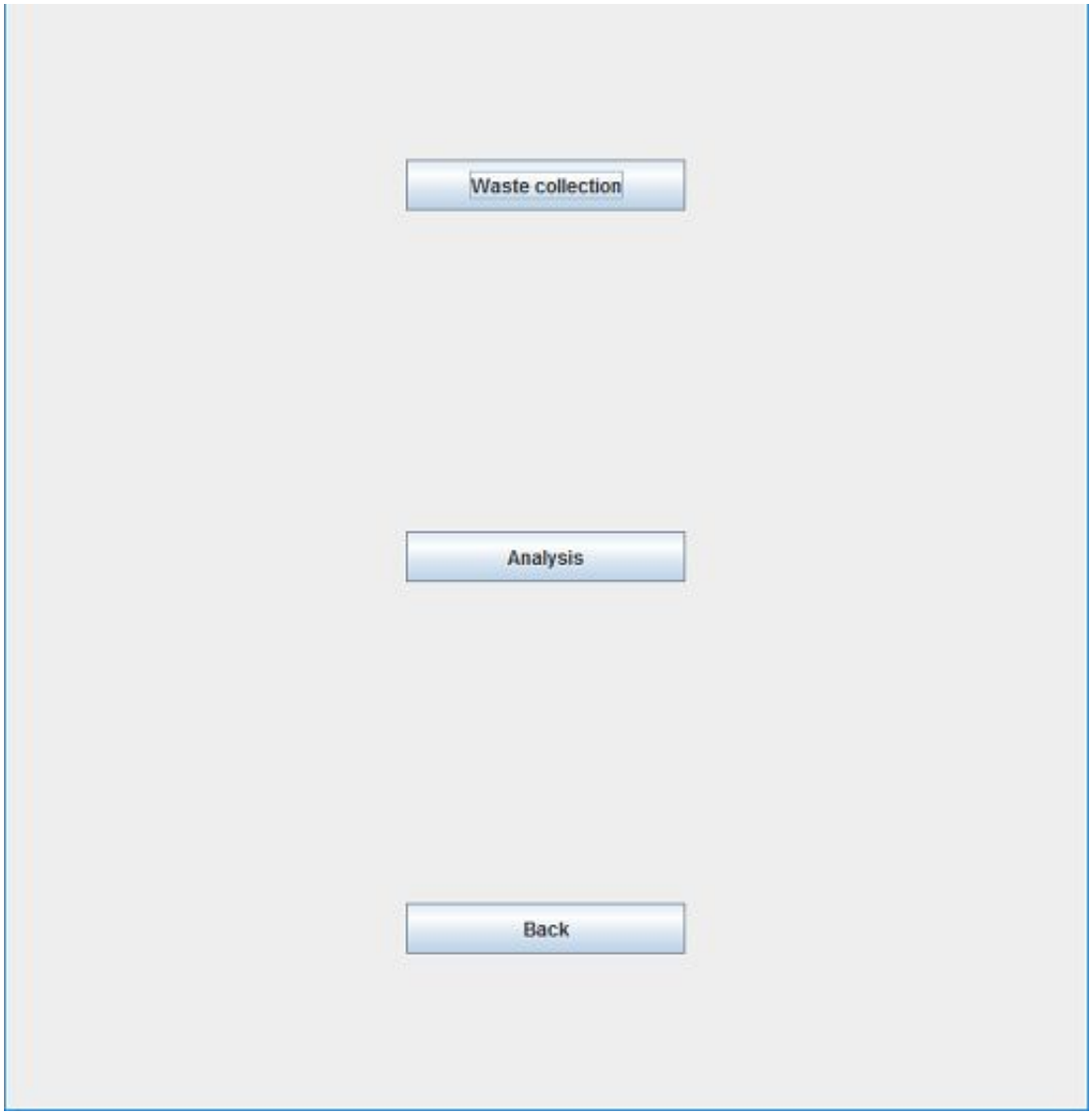


Na opção “Waste recovery” pode-se analisar o mapa da região e as soluções obtidas pelos algoritmos.

Na opção “Options” pode-se escolher os ficheiros do mapa da região e informações dos contentores, assim como customizar os valores do alfa e do beta a utilizar no algoritmo A*.

O botão “Exit” fecha a aplicação.

Menu “Waste recovery”:

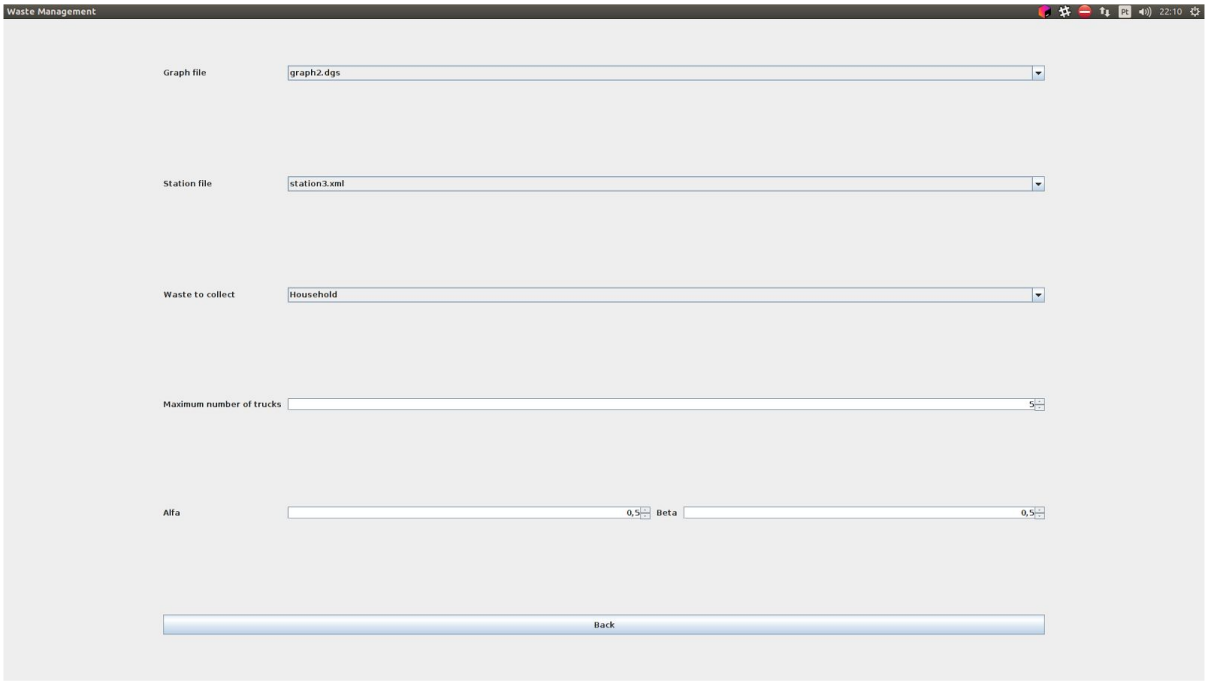


Botão “Waste Collection”: Permite ver e comparar as soluções obtidas.

Botão “Analysis”: Permite obter informações sobre o lixo existente em cada contentor, e a capacidade dos camiões.

Botão “Back”: Permite voltar ao menu principal.

Página das “Options”:

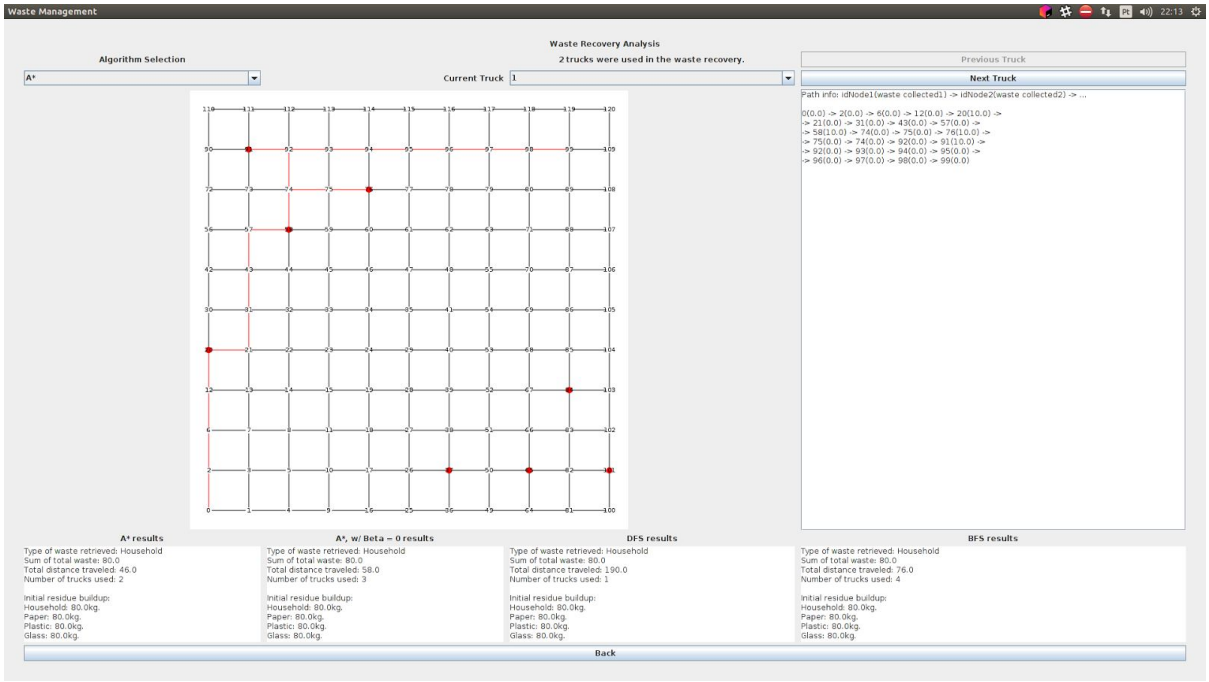


Nesta página é possível escolher o ficheiro do grafo que representa o mapa da região, o ficheiro da estação que informa sobre a capacidade dos camiões e contentores.
Além disso, é possível escolher o número máximo de camiões que serão utilizados pelo algoritmo.

Também pode-se se escolher o lixo a recolher: “Household”, “Paper”, “Glass” e “Plastic”.

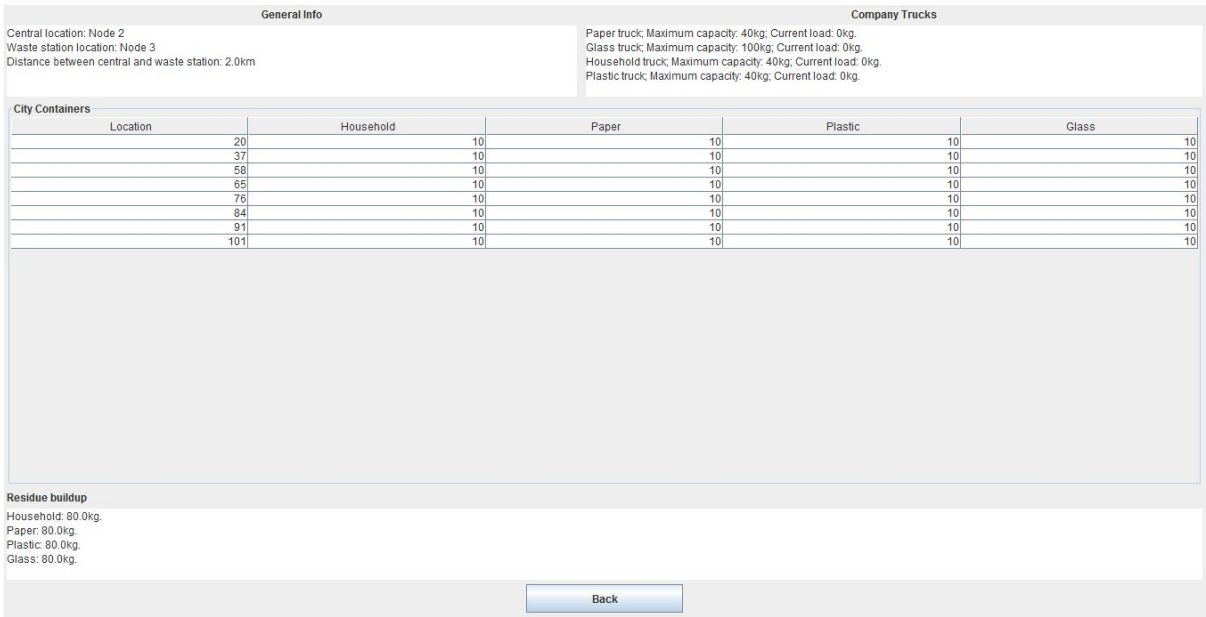
Como dito anteriormente, os valores Alfa e Beta utilizados pelo A* podem ser editados. A soma de alfa e beta tem que ser igual a 1. Quando Alfa é maior dá-se mais privilégio à distância percorrida (quanto menor melhor), e quando Beta é maior dá-se mais privilégio ao lixo recolhido.

Página “Waste collection”:



Aqui pode-se analisar as soluções dos vários algoritmos utilizados bem como o caminho percorrido por cada camião e o lixo recolhido em cada nó. Por último, pode-se verificar também os resultados finais de cada algoritmo.

Página “Analysis”:



Location	Household	Paper	Plastic	Glass
20	10	10	10	10
37	10	10	10	10
58	10	10	10	10
65	10	10	10	10
76	10	10	10	10
84	10	10	10	10
91	10	10	10	10
101	10	10	10	10

Contém informações sobre os contentores e camiões, sendo possível editar o lixo atual de um determinado contentor.