Universidade do Porto

FEUP Faculdade de Engenharia

Mestrado Integrado em Engenharia Informática e Computação

Inteligência Artificial

3º Ano

2º Semestre

Otimização da Navegação numa Cidade

Turma 4, Grupo 24

Daniel Filipe Silva Ermida Martins de Freitas – 090509091 – ei09091@fe.up.pt

Tiago Manuel de Castro Rodrigues – 090509042 – ei09042@fe.up.pt

Tiago Marques Dias da Mota – 090509068 – ei09068@fe.up.pt

Data de entrega: 16-04-2012

Índice

Objetivo

O presente trabalho surge no âmbito da unidade curricular de Inteligência Artificial, cujo o objetivo, face ao trabalho proposto, é a resolução de um problema de otimização. Esta resolução deverá passar pela implementação de algoritmos que a permitam efetuar, versando essencialmente num, de entre três possibilidades apresentadas: Algoritmos Genéticos, Pesquisa Tabu ou Arrefecimento Simulado.

O relatório final que se apresenta pretende descrever a estruturação do trabalho feito, assim como, de que forma foi implementado o algoritmo que irá permitir a resolução do problema de otimização proposto. Optou-se pela escolha dos Algoritmos Genéticos, divergindo do apresentado no relatório intermédio, mas abordado e justificado perante o docente da unidade curricular.

O problema alvo de estudo e implementação é a determinação do percurso por um automobilista, para se deslocar numa cidade, passando por um conjunto de locais a especificar pelo utilizador. O automobilista desloca-se num carro que até ao final do percurso não deverá gastar o seu depósito de combustível e deverá sempre percorrer as paragens definidas pelo utilizador, num ambiente de uma cidade que poderá também ela ser editável. Todo o problema assenta na premissa de que o percurso terá de ser efetuado no menor tempo possível.

Para tal, recorrendo à linguagem de programação Java e ao algoritmo supracitado, pretende-se procurar a melhor forma de resolver o problema em questão.

Descrição

Pretende-se desenvolver uma aplicação em Java que devolva o melhor caminho a percorrer, no menor tempo possível para um automobilista que tem de efetuar um percurso especificado pelo utilizador numa cidade, passando por vários pontos no mapa que representam locais de interesse do género (Casa, Café, Bomba de Gasolina, Tabacaria, Escola, etc.), tendo como paragem e objetivo final, deslocar-se até ao escritório. De referir que o ponto inicial especificado é sempre a casa do utilizador, identificada no mapa com o número 1. A cidade é representada por um grafo, onde os nós representam os pontos de interesse e as arestas as ruas, que podem ser de sentido único ou não, que ligam esses pontos. Atenta-se ainda no fato de que nos dois sentidos, o percurso não ter necessariamente a mesma distância, devido à existência das ruas de sentido único, havendo necessidade de tomar caminhos mais longos para se atingir um determinado ponto. Considera-se um mapa base ao qual poderão ser adicionados pontos de interesse que se ligam por novas ruas aos já existentes, tornando o mapa da cidade editável. Desta forma, garante-se sempre uma base inicial de trabalho sem a qual não será possível ter a aplicação a funcionar com os resultados espectáveis.

A ideia da aplicação é retornar o percurso especificado, levando o menor tempo possível a percorrer, garantindo sempre que o automóvel tem combustível para chegar ao destino e em caso de necessidade poder abastecer na bomba mais próxima. Desta forma, o programa deverá garantir que o automobilista chega ao destino e nunca fica sem combustível. O percurso deverá sempre começar na Casa do condutor e tem sempre como objetivo final chegar ao Escritório.

No sentido de calcular o menor tempo possível, a cada rua, para além da distância da mesma, mantemos também a velocidade máxima permitida na mesma, de tal forma que seja possível calcular qual o tempo que o automobilista demora a efetuar o percurso, sendo este cálculo uma aproximação da realidade, sem ultrapassar a velocidade máxima permitida.

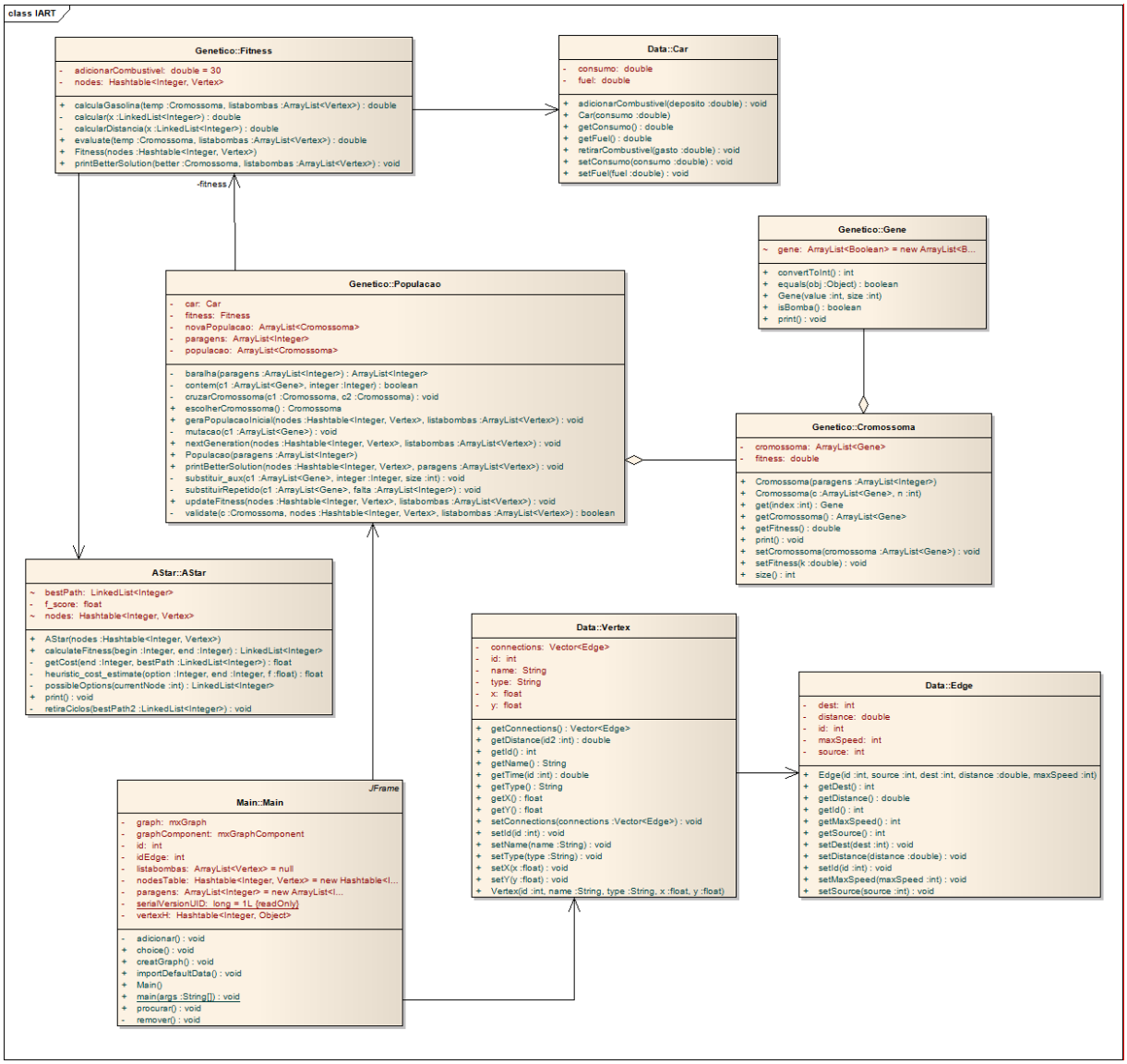
Funcionalidades

O programa oferece duas principais funcionalidades:

- Retorno do caminho desde o ponto inicial até ao Escritório (ponto final), passando pelos pontos de interesse definidos pelo utilizador.

- Edição do mapa da Cidade, ou seja, adicionar ou remover pontos de interesse da Cidade, assim como as estradas que os ligam.

Estrutura do programa



Neste projeto é feita uma divisão em 4 módulos:

Main - Módulo e classe principal do programa. A partir desta são chamados os métodos necessários e responsáveis pela aplicação devolver o resultado melhor para o caminho imposto pelo utilizador.

Data - Módulo responsável por guardar os dados relativos à Cidade (pontos de interesse e ruas) e também a informação do Carro (número de Km que o carro ainda consegue fazer, de acordo com a gasolina que tem).

Genético - Toda a implementação e posterior utilização do algoritmo genético no programa, provém deste módulo. Através de cromossomas que representam um caminho, as classes aqui presentes, aplicam o algoritmo e devolvem o resultado esperado.

AStar - Módulo onde reside a implementação do algoritmo de pesquisa A*. Cada cromossoma representa um conjunto de genes, que por sua vez representam paragens a efetuar pelo carro. A aplicação do algoritmo no programa é a um nível mais específico que o genético, uma vez que é apenas usado para encontrar o caminho mais curto entre duas paragens dadas.

Esquemas de Representação de Conhecimento

Numa análise inicial ao enunciado do trabalho proposto, a primeira abordagem que se pensou utilizar passaria pela implementação do algoritmo de Arrefecimento Simulado. Em discussão com o docente das aulas práticas, chegou-se à conclusão que provavelmente seria de maior interesse para o grupo a resolução do problema recorrendo à implementação de algoritmos genéticos.

Novo estudo foi efetuado, mas sem grandes resultados no que toca a uma implementação básica de algoritmos genéticos. Desta forma, associaram-se dois algoritmos inteligentes de tal forma que se procurasse uma solução mais viável e eficaz de acordo com o problema proposto. Para além da implementação do algoritmo genético de caráter evolutivo, associou-se a utilização

do algoritmo A* para proceder à resolução do problema. Segue-se a explicação detalhada dos procedimentos efetuados:

Em primeiro lugar, no âmbito dos algoritmos genéticos, gera-se aleatoriamente uma população inicial de 30 indivíduos que são todos válidos para o problema em questão. A estrutura dos cromossomas é [1, S/N, paragem, S/N, paragem, …, 2], onde paragem é uma paragem obrigatória especificada pelo utilizador e o S/N é a necessidade que o utilizador tem de se deslocar à bomba para abastecer entre cada ponto de paragem. Sim é especificado com 1 e Não com 0. Na geração aleatória, as verificações existentes relacionam-se com a existência no cromossoma das paragens obrigatórias introduzidas pelo utilizador e nunca com a necessidade de ir à bomba abastecer, o que permite uma maior geração de cromossomas distintos que se traduz em voltas desnecessárias pelo mapa da cidade. São estes dados que irão converter para o melhor resultado possível ao fim de um determinado número de gerações, encontrando assim o melhor caminho em termos de tempo.

Seguindo ainda a nível do algoritmo genético, a fase que se segue à geração da população inicial é a avaliação dos cromossomas, utilizando a denominada de função de fitness. O objetivo desta etapa é avaliar os cromossomas, determinando os melhores. Assim, a função de fitness avalia o caminho representado no cromossoma em função do tempo gasto no percurso. Sendo o mapa um grafo pesado com as distâncias entre os pontos nas arestas, o tempo que demora a percorrer cada aresta é o valor da distância dividido pela velocidade máxima permitida na aresta. A soma dos valores dos tempos entre os vários pontos resulta nos valores da função de fitness, significando que quanto menor for o valor obtido, melhor é o caminho percorrido. Nesta fase entra a utilização do A*. Em cada ponto de paragem é calculado o melhor caminho para atingir o ponto de paragem obrigatório seguinte, passando por pontos intermédios não relevantes para o estudo do problema. Para além disto, verifica se o depósito de combustível é suficiente para atingir a paragem seguinte, ou se é necessário ir à bomba de gasolina abastecer. Sempre que o A* define um caminho inútil, do género 4 - 8 - 4, ou seja, uma ida a um ponto onde é obrigatório efetuar um retorno, esse caminho é descartado.

Na etapa de cruzamento, definiu-se uma probabilidade de cruzamento de 80%, assim, dos 30 indivíduos que existem na população, apenas há probabilidade de 80% de se cruzarem, havendo maior probabilidade de se cruzarem aqueles que tiverem o melhor valor de fitness, no caso presente, o menor tempo. Para a escolha aleatória dos cromossomas a cruzar, mantendo o

pressuposto de que os melhores têm maior probabilidade de se cruzar, utilizou-se o método da roleta direta, onde os valores utilizados foram, não os tempos, mas o quociente entre 1000 e os tempos, o que garante a mesma proporcionalidade na aplicação do método da roleta direta onde, desta forma os valores mais altos têm maior probabilidade de serem escolhidos. Assim, escolhem-se pares de cromossomas para emparelhar, originando novos cromossomas. O cruzamento faz-se num ponto aleatório do cromossoma, originando dois cromossomas novos. Um dos cromossomas fica com a informação à esquerda do ponto de cruzamento do primeiro e a informação à direita do segundo. O outro fica com a restante informação que é exatamente o oposto do primeiro, isto é, a informação à direita do primeiro e a informação da esquerda do segundo. Isto repete-se para 80% da população, originando no caso concreto, 24 novos cromossomas. Estes novos cromossomas sofrem um processo de validação novo para saber se fazem sentido existir na população em questão. Assim, a verificação efetuada prende-se com a constatação da existência de todas as paragens obrigatórias definidas pelo utilizador. Caso isto não se verifique, significa automaticamente que existem paragens repetidas. Assim, avalia-se o cromossoma em questão e substitui-se aleatoriamente as posições repetidas pelas posições em falta, originando assim um cromossoma válido para a população.

Neste ponto, surge a maior decisão da utilização da genética do algoritmo. Aplicado ao problema em questão, optou-se por utilizar o algoritmo genético com “overlapping populations”, isto é, depois de produzir o cruzamento, analisam-se os 24 + 30 cromossomas, ordenando-os pelo melhor fitness, descartando os 24 piores, ficando apenas com 30 cromossomas que serão os constituintes da próxima geração. A utilização de mutação não se revelou vantajosa neste algoritmo dado que iria introduzir uma aleatoriedade que pareceu desinteressante dada a resolução do problema em questão. Assim, constitui-se a população para a nova geração e o processo repete-se ao longo de 10.000 gerações, obtendo-se o resultado pretendido.

Análise da complexidade dos algoritmos que usou

No projeto são utilizados dois algoritmos:

Algoritmo genético:

Este algoritmo tem uma complexidade espacial O(2n), porque o espaço de memória utilizado aumenta sempre em dobro dependendo do número de paragens obrigatórias que o

utilizador definir. A cada paragem está associada um valor que é o que indica a necessidade ou não de ir abastecer à bomba, daí a complexidade espacial ser definida desta forma.

Em termos de complexidade temporal, revela-se alguma ineficiência, demarcada na principalmente na função de cruzamento dos cromossomas. Neste método é necessário procurar os genes que estejam repetidos de entre as paragens existentes, saber quais as paragens que faltam e por fim fazer a substituição de forma aleatória. Em termos de implementação isto gera 3 ciclos for encadeados, originando no pior caso uma complexidade temporal de O(n^3).

A*:

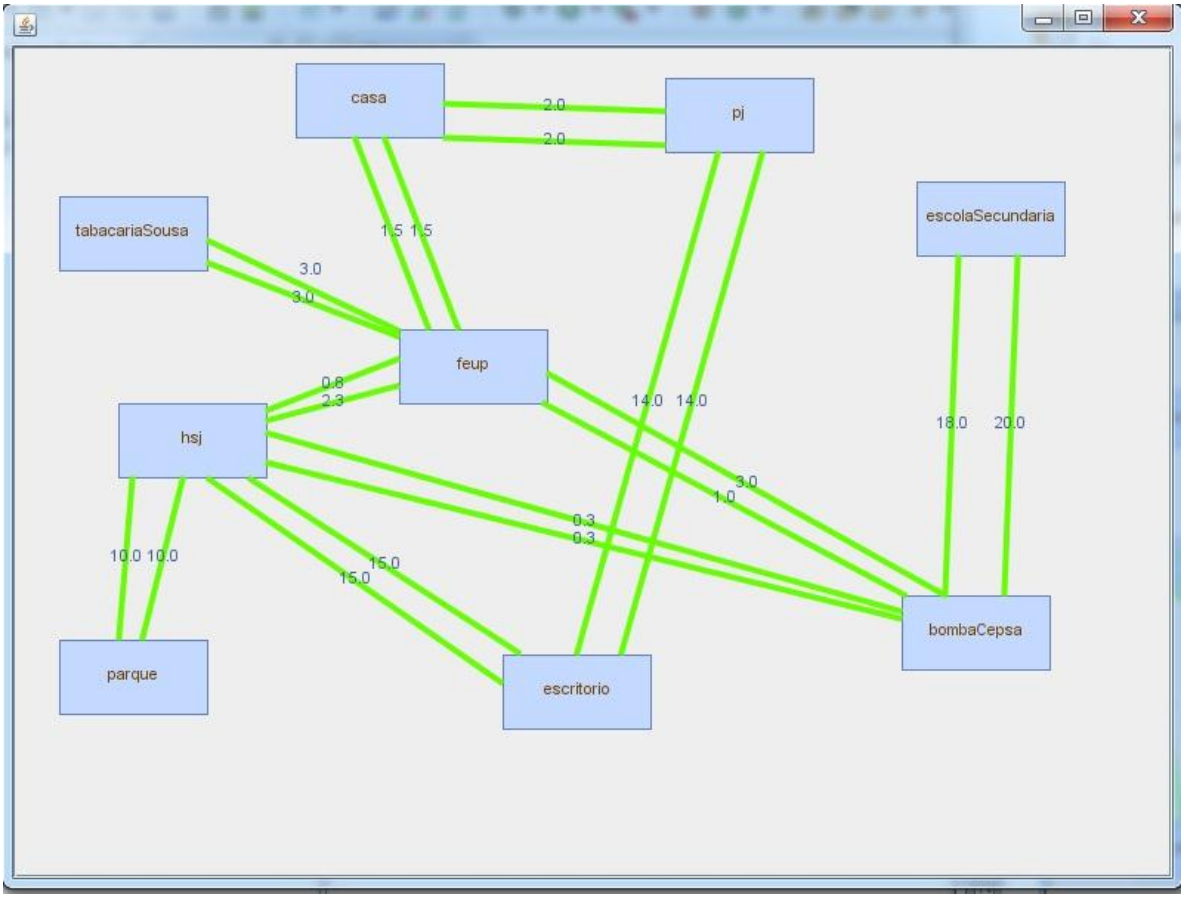
Em termos de complexidade espacial esta é O(2n) visto que o A* mantém duas listas com as paragens visitadas e as que ainda estão por visitar. Desta forma, o aumento das paragens obrigatórias reflete-se num aumento duplo.

A complexidade temporal é 2 ^ $O(n)$, no pior caso, sendo exponencial ao número de ligações de arestas que existam entre os nós do grafo, verificados que são duplamente e inseridos nas listas de visitados ou a visitar.

Ambiente de desenvolvimento

O projeto foi desenvolvido e implementado em Eclipse, utilizando a linguagem de programação Java 1.6. O seu sistema operativo usado foi o Windows 7, versão 64-bits.
Relativamente ao software extra, utilizamos apenas a biblioteca "jgraphx.jar", que permite ao utilizador visualizar o mapa da cidade, representada por um grafo onde os nós são os pontos de interesse e as arestas, as ruas que os ligam.

Avaliação do programa



Simulações

1) Partir de casa, passando na feup e hsj, terminando no escritório. Analisando o mapa da cidade podemos constatar que o melhor percurso é casa->feup->hsj->escritório.

O resultado do programa é o mesmo que o esperado, como se comprova com a imagem seguinte.

```
Ponto de Partida:1
Pontos obrigatórios de passagem (separados por virgulas):4,5
Inicio do processo
Populacao inicial gerada
Fim do processo
Percurso até à proxima paragem: casa->feup
Percurso até à proxima paragem: feup->hsj
Percurso até à proxima paragem: hsj->escritorio
```

2) Partir de casa, com paragens na feup, tabacariasousa e parque, terminando o seu percurso no escritório. Analisando o mapa da cidade podemos constar que tem de seguir casa->feup->tabacariaSousa->feup->bombaCepsa->hsj->parque->hsj->escritório. É isto que o programa também retorna.

```
Ponto de Partida:1
Pontos obrigatórios de passagem (separados por virgulas):4,8,9
Inicio do processo
Populacao inicial gerada
Fim do processo
Percurso até à proxima paragem: casa->feup
Percurso até à proxima paragem: feup->tabacariaSousa
Percurso até à proxima paragem: tabacariaSousa->feup->bombaCepsa
Percurso até à proxima paragem: bombaCepsa->hsj->parque
Percurso até à proxima paragem: parque->hsj->escritorio
```

Conclusões

O relatório final espelha de forma clara e sucinta o trabalho desenvolvido pelo grupo de trabalho, face ao tema proposto pelos docentes.

Inicialmente, ficou decidido pelo grupo utilizar um algoritmo de "Arrefecimento Simulado", no entanto, após a apresentação do relatório intermédio e consequente "feedback" da docente das aulas práticas, optou-se pela utilização de um algoritmo genético. Pelas razões referidas ao longo do relatório, esta implementação foi bastante difícil e morosa, exigindo que fosse feita em dois níveis: Primeiro - Genético, Segundo - A*. Devido a isto, não foi possível realizar a interface gráfica proposta inicialmente, fazendo com que o programa não disponha da usabilidade indicada para um utilizador comum.

Hoje em dia, muitas das máquinas (computadores) utilizam processadores multi-core sendo que a implementação de um sistema "multithreading" poderia melhorar bastante a rapidez de execução do programa, visto que, assim, seria possível dividir os cálculos a serem efetuados pelo algoritmo, pelos "cores" existentes no processador, em vez de ser apenas um deles a efetuar todos (imensos) os cálculos.

Quanto à opção de usar ArrayList, por vez de HashTable, para guardar a informação de Cromossomas e Genes, esta foi tomada com base no facto de a cidade inicial apresentada ser

pequena, o que faz com que o tamanho das ArrayLists usadas ser, também, pequeno. Portanto, como HashTables contêm um grande *overhead*, apenas compensaria usar esta estrutura de dados, se a cidade já tivesse um tamanho considerável, evitando assim ArrayLists com tamanhos elevadissímos.

Recursos

Bibliografia:

Genetic Algorithm Tutorial - Part 2,

"http://www.leolol.com/drupal/tutorials/theory/genetic-algorithms-tutorial-part-2-computer-theory"

Página da Disciplina: Métodos de Resolução de Problemas e Algoritmos para Evolução, "http://paginas.fe.up.pt/~eol/IA/1112/APONTAMENTOS/2MRPeAG.pdf"

Wikipédia: A* search algorithm,

"http://en.wikipedia.org/wiki/A*_search_algorithm"

Anexos

Manual do utilizador

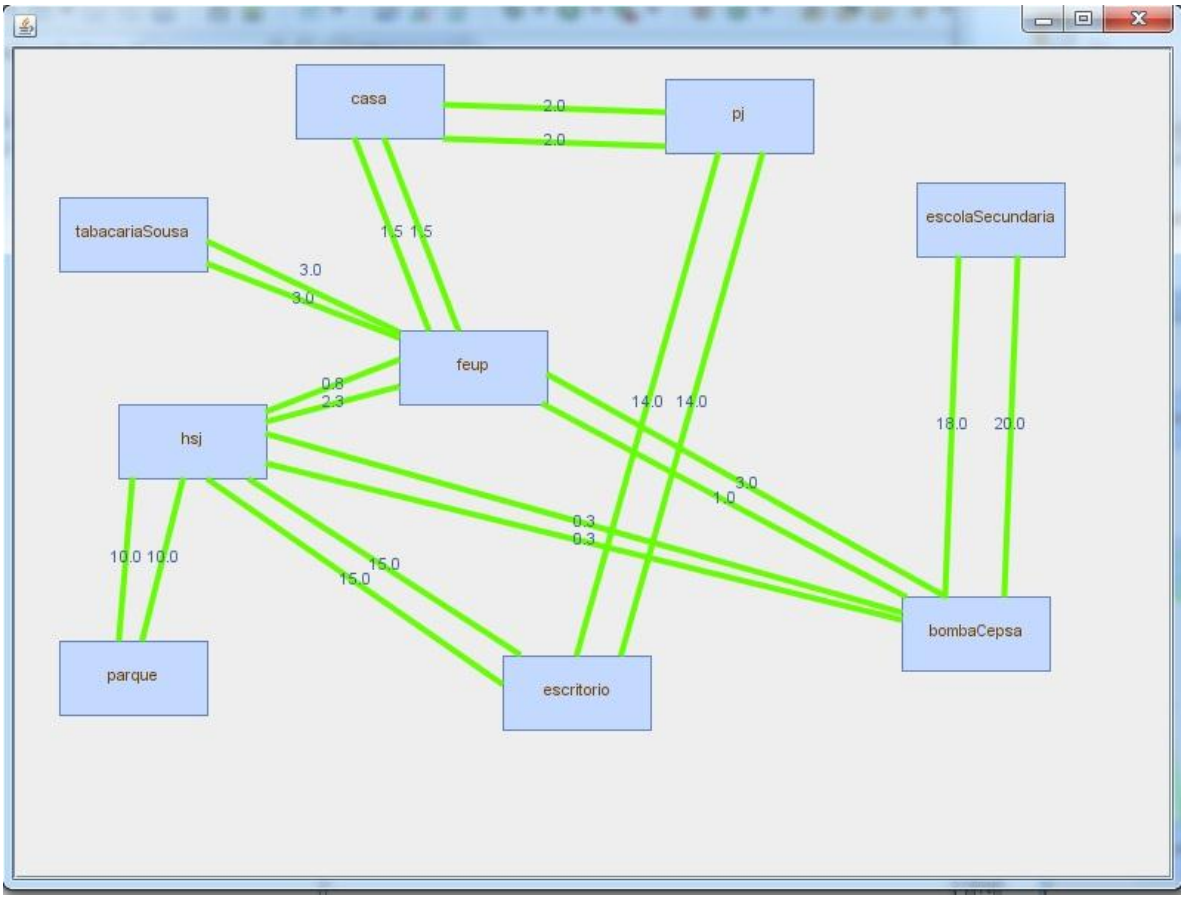
Antes de executar o projeto é necessário adicionar às bibliotecas externas o jgraphx.jar, que se inclui na pasta submetida, juntamente com o código.

No início do programa são apresentadas ao utilizador 3 opções, procurar, adicionar nó, remover nó.

O nas duas últimas opções o utilizador tem que inserir dados para a adição e remoção de nós.

Na opção de procura o utilizador tem que inserir o nó de partida, de seguida todas as passagens obrigatórias. E o programa retornará a melhor opção de ligação da origem até ao escritório passando por todas as paragens anteriormente referidas

Exemplo de uma execução




```
Ponto de Partida:1
Pontos obrigatórios de passagem (separados por virgulas):4,5
Inicio do processo
Populacao inicial gerada
Fim do processo
Percurso até à proxima paragem: casa->feup
Percurso até à proxima paragem: feup->hsj
Percurso até à proxima paragem: hsj->escritorio
```

Na primeira situação o ponto de partida é a casa, com passagem na feup e no hjs, terminando o seu percurso no escritório.

```
Ponto de Partida:1
Pontos obrigatórios de passagem (separados por virgulas):4,8,9
Inicio do processo
Populacao inicial gerada
Fim do processo
Percurso até à proxima paragem: casa->feup
Percurso até à proxima paragem: feup->tabacariaSousa
Percurso até à proxima paragem: tabacariaSousa->feup->bombaCepsa
Percurso até à proxima paragem: bombaCepsa->hsj->parque
Percurso até à proxima paragem: parque->hsj->escritorio
```

Na segunda situação o ponto de partida é a casa, com paragens na feup, tabacariasousa e parque, terminando o seu percurso no escritório.

```
Ponto de Partida:4
Pontos obrigatórios de passagem (separados por virgulas):7,8,9
Inicio do processo
Populacao inicial gerada
Fim do processo
Percurso até à proxima paragem: feup->tabacariaSousa
Percurso até à proxima paragem: tabacariaSousa->feup->bombaCepsa->escolaSecundaria
Percurso até à proxima paragem: escolaSecundaria->bombaCepsa
Meteu Gasolina
Percurso até à proxima paragem: bombaCepsa->hsj->parque
Percurso até à proxima paragem: parque->hsj->escritorio
```

Na terceira situação o ponto de partida é a feup, com paragens no parque, tabacariaSousa e escolaSecundaria, terminando o seu percurso no escritório.