Relatório Trabalho Prático - IA - Fase 2

Grupo 32:
António Luís de Macedo Fernandes (a93312)
José Diogo Martins Vieira (a93251)
João Silva Torres (a93231)
Ricardo Lopes Santos Silva (a93195)

Jan 5, 2022

Contents

1 Introdução

No âmbito do desenvolvimento da segunda fase do projeto da unidade curricular Inteligência Artificial foi-nos proposto atualizar o sistema anteriormente desenvolvido, criando um sistema de recomendação de circuitos de entrega de encomendas.

Dito isto, criamos um cenário realista, correspondente a parte da cidade de Braga, onde inserimos não só a nossa empresa de distribuição Green Distribution como também todas as freguesias às quais os estafetas conseguem chegar. Este está representado sob a forma de um grafo que será explicado detalhadamente mais à frente.

Ao longo deste relatório iremos também iremos abordar o a forma como desenvolvemos a base de conhecimento, as queries propostas, os algortimos de procura não informada e informada e, por fim, uma análise de resultados onde comparamos os algortimos de procura.

Na última secção do relatório faremos uma análise crítica sobre toda a projeção e desenvolvimento do trabalho.

2 Formulação do Problema

Através de uma leitura inicial do enunciado, entendemos que teríamos que criar um cenário, onde seriam efetuadas as entregas.

Visto isto, decidimos que o nosso grafo ia ser constituído por várias freguesias de Braga, mais precisamente, as freguesias em redor de Gualtar, onde determinamos que seria o nosso armazém de distribuição da *GreenDistribution*(Centro). Então, através das distâncias reais e das distâncias dos trajetos mais eficientes, fornecidos pelo *Google Maps*, construímos o *grafo* para o nosso problema. Os nodos correspondem às freguesias e as arestas às distâncias, correspondestes aos trajetos mais curtos, que o estafeta terá de percorrer entre elas. Dentro de cada nodo também se encontram as estimativas de custo até às restantes localidades do grafo.

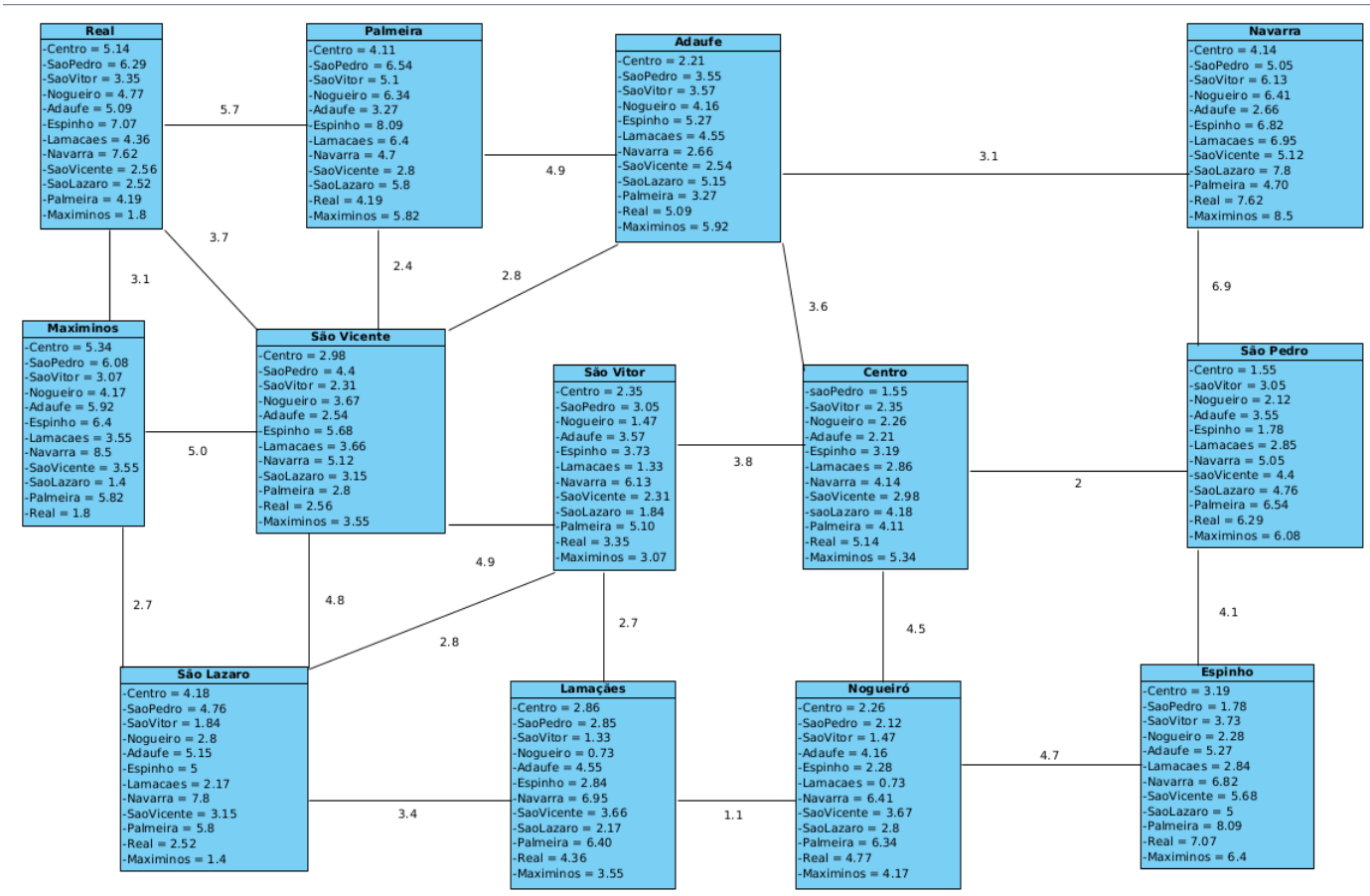

Figure 1: Grafo.

Estado inicial: Local onde será iniciada a entrega - centro de distribuição *GreenDistribution*.

Estado final: Destino da encomenda a ser entregue.

Operadores: deslocação do estafeta para outra freguesia, tem com pré-condição essa freguesia ainda não ter sido visitida.

Custo da Solução: Distância percorrida até chegar ao destino, custo das arestas do caminho entre o centro e o local de entrega.

3 Base de Conhecimento

Nesta segunda fase, definimos todas as arestas e nodos existentes no nosso grafo. Cada aresta é definida por aresta(nó1, nó2, custo), sendo o custo a distância do trajeto entre o centro e a freguesia correspondente. Cada nodo é definido por nodo(Destino,nó, estimativaDoCusto: nó->Destino), isto é, custo aproximado do nó até ao destino.

Alteramos o prazo limite na base de conhecimento da primeira fase para facilitar o acesso à mesma na segunda fase ao verificar se um veículo realiza a entrega a tempo (prazolimite(Id,Data,????)).

Também recorremos à base de conhecimento definida para a primeira fase:

- encomenda(id, artigo, morada, preço, peso, volume,data,tempo);
- entrega(idEncomenda,cliente, estafeta, meioTransporte, preçoEntrega, prazolimite, rank,date,time);
- ecologico(veiculo, nr_ecologico);
- meioTransporte(Veiculo,velMedia,PesoMax);

4 Queries

4.1 Gerar os circuitos de entrega, caso existam, que cubram um determinado território (e.g. rua ou freguesia)

Esta query retorna todos os caminhos do local de partida até ao destino, sendo-lhes associado o seu custo total.

Para tal, o processo utilizado passa por obter os nodos adjacentes do atual e verificar se estes não se encontram no caminho, de modo a evitar ciclos, repete-se o processo até se ter chegado ao nodo de destino. Após isso verifica-se o custo do caminho, e devolve-se um par com o caminho e o seu custo.

4.2 Representação dos diversos pontos de entrega em forma de grafo, tendo em conta que apenas se devem ter localizações (rua e/ou freguesia) disponíveis

Cada nodo corresponde a uma freguesia, e as suas arestas correspondem no caminho mais curto entre estes. Cada nodo tem também uma estimativa da distância para cada nodo do grafo, que corresponde à distância em linha reta entre os nodos.

4.3 Identificar quais os circuitos com maior número de entregas (por volume e peso)

Para esta query, decicimos devolver o top 3 de circuitos com maior número de entregas por peso e volume, começamos por usar um findall que nos permite verificar todas as encomendas que já foram entregues, e obter informações relativamente essa encomenda, nomeadamente: a morada, o peso e o volume, criando assim uma lista de triplos.

De seguida, queremos agrupar os pesos e os volumes para a mesma morada, que é feito na função: $agrupaPorMorada(Lista, Res)$, que vai percorrer a Lista, e para cada elemento chama a função: $countAllPesoVolume$, que recebe a morada do elemento, e a Lista, e vai devolver a soma do peso e volume para morada dada, assim como uma nova lista sem os elementos que contêm a morada, deste modo evita-se percorrer a lista toda novamente, percorrendo apenas os elementos que ainda não foram agrupados. Assim é construída uma lista de pares Morada-Soma, sendo a Soma, o cálculo de todos os valores peso e volume das encomendas entregues associadas àquela morada.

Após isto, é usado a função pré-definida no PROLOG *sort*, que vai organizar a lista de forma decrescente. Pegamos nos primeiros três elementos da lista, e transformamos a morada no circuito correspondente. Assim, a o resultado é compostao por uma lista com no máximo 3 pares (Circuito-Soma).

De um modo geral, alcançámos todos os objetivos propostos e conseguimos, utilizando um cenário mais real, verificar como estes tipos de pesquisa atuam na nosssa base de conhecimento.

Desta forma, através da realização deste trabalho,

4.4 Comparar circuitos de entrega tendo em conta os indicadores de produtividade

Primeiramente, definimos os decréscimos velocidades por cada Kg da encomenda transportada através de um determinado meio de transporte: velocidadeEntrega(MeioTransporte,Peso,Res) - Dado um MeioTransporte e um peso, devolve o resultado do cálculo consoante o decréscimo, que corresponde ao tempo total necessário para a entrega, em horas.

De seguida, queremos obter os veículos capazes de realizar uma encomendas, tendo em atenção o prazo limite e o peso máximo da encomenda. Para tal foi definida uma função para esse propósito $getVeiculoParaEntrega(Id, Veiculo, Time)$, que para um determinado Id de uma encomenda, vai devolver o Veículo capaz de a realizar juntamente com o tempo que demora a realizar, em horas. Verifica se o peso da encomenda é permitida para os diversos meios de transporte, caso seja, obtemos a velocidade de entrega para o meio de transporte explicada acima, o custo do caminho até ao destino. E calculamos, o tempo que o meio de transporte iria demorar, verificando assim se esse tempo é menor que o prazo limite.

Assim dado um id de uma encomenda, realizamos um findall da função $getVeiculoParaEntrega$ para obter uma lista com pares (MeioTransporte,Horas), que será transformada num par (MeioTransporte,Minutos) para um resultado mais claro. Finalizamos com o uso da função *sort*, para ordenar os restados de forma crescente de tempo necessário para realizar a entrega.

4.5 Escolher o circuito mais rápido (usando o critério da distância)

Nesta query é pedido o circuito mais rápido, portanto consideramos que todas as entregas são feitas de carro, visto que, apesar de ser o meio de transporte menos ecológioco, é o mais rápido.

Dito isto, desenvolvemos a função $caminhoMaisRapidoDistancia(Id, H, carro)$ que recebe como argumentos o Id da encomenda, a variável que, no final, vai conter o circuito do return e, por fim, o carro que é, como foi dito no parágrafo anterior, o meio de transporte pré-definido para esta consulta.

Ora, a nossa função verifica se existe uma encomenda com o Id passado como argumento e com um Destino. Consequentemente, através de um findall, agrupa numa lista todos os caminhos para esse Destino. Finalmente, ordena esta lista por ordem crescente, com um sort, e coloca a cabeça da lista na variável de return (que correponde ao caminho mais curto).

4.6 Escolher o circuito mais ecológico (usando um critério de tempo)

Nesta query é pedido o circuito mais ecológico, ou seja, teremos que verificar se no limite de tempo definido os veículos, por ordem decrescente de número ecológico, conseguem cumprir o circuito.

Definimos, então, a função $caminhoMaisEcologicoTempo(Id, Res)$ em que o Id trata-se do id da encomenda e o Res será a lista de veículos que conseguem efetuar o circuito no tempo limitado por ordem crescente. Recorremos então à função $circuitoVeiculoTempoEntrega$ explicada anteriormente. Por fim, é usada o *sort* de modo a ordenar a lista de menor para o maior tempo.

5 Algoritmos de Procura

Nesta secção iremos abordar os vários tipos de procuras informadas e não informadas.

5.1 Procura não informada

Este tipo de procura, ao contrário da informada, é utilizada quando apenas nos fornecem a definição do problema sem qualquer passo para a procura da solução. Encontram soluções para problemas pela geração sistemática de novos estados, que são comparados com o objectivo. No entanto, estes métodos ,na maioria dos casos, são ineficientes pois são métodos exaustivos.

Os vários tipos de procura não informada abordados foram:

Note que:

b: o máximo fator de ramificação (o número máximo de sucessores de um nó) da árvore de pesquisa;

d: a profundidade da melhor solução;

m: a máxima profundidade do espaço de estados.

5.1.1 Profundidade (DFS - Depth-First Search)

O algoritmo de procura em profundidade tem como característica base a expansão do primeiro nó filho da árvore de busca até que o alvo da procura seja encontrado ou que se depare com um nó sem filhos. Seguidamente, a busca retrocede (backtracking) e recomeça no seguinte nó.

A nível temporal, este algoritmo pode ser mais demorado quanto maior for o grafo, ou seja, não é um algoritmo eficiente caso se esteja a lidar com grafos extensos e complexos. No entanto, é útil nos casos em que o problema apresenta várias soluções.

A complexidade espacial desta procura trata-se do comprimento do caminho mais longo e por cada nodo terá de se guardar os vizinhos, ou seja, para cada m nodos terá que se guardar os b nodos extra.

5.1.2 Largura (BFS - Breadth-First Search)

O algoritmo de procura em largura realiza uma busca exaustiva num grafo passando por todos os vértices e arestas, isto é, o algoritmo desta pesquisa garante que nenhuma aresta é visitada mais do que uma vez. Este tipo de pesquisa em largura é frequentemente utilizado na travessia ou busca num grafo.

A pesquisa funciona inicialmente num nó raiz e realiza a busca pelos seguintes nós vizinhos e assim sucessivamente, percorrendo todos os nodos inexplorados. Por este motivo, a pesquisa feita é muito sistemática e completa. Os pontos negativos serão então a grande carga de memória utilizada e o elevado período de tempo para realizar este tipo de pesquisa.

Pelo que foi dito anteriormente, este algorito BFS é mais válido em travessias curtas de grafos com menor número de nodos. Em casos em que o custo de travessia de uma aresta seja

1, este algoritmo consegue ser bastante ótimo e produzir melhores resultados. Para além disso, se se pretender procurar um elemento próximo da raiz de uma arvore, este método será mais eficiente. Em contrapartida, se se deparar com uma árvore muito extensa ou procurarmos uma solução no fim desta, BFS torna-se bastante impraticável devido ao elevado uso de memória.

Complexidade Temporal: $O(b^d)$;

Complexidade Espacial: $O(b^d)$.

5.1.3 Busca Iterativa Limitada em Profundidade

A busca em profundidade iterativa procura combinar as virtudes da busca em profundidade e da busca em largura. São elas a relativa pouca necessidade de memória e a capacidade de examinar todo o espaço de estados encontrando a solução ótima, respetivamente.

Na primeira iteração, o grafo é gerado utilizando a busca em profundidade limitada com limite igual a 1. Se a solução não foi encontrada, inicia-se a segunda iteração: todo o grafo anterior é descartado e um novo é construído através da busca em profundidade limitada com limite igual a 2. Se ao final da segunda iteração a solução ainda não tiver sido obtida, o processo continua definindo o limite igual a 3 e assim sucessivamente. A busca pára quando a solução é encontrada ou quando toda o grafo for gerado.

No entanto, caso o limite estabelecido não seja a profundidade suficiente para encontrar a solução esta procura revela-se incompleta. A vantagem desta procura comparada à DFS é que resolve o problema do loop infinito caso o grafo tenha profundidade infinita.

Ao implementar esta procura temos como argumento a profundidade limite até à qual é possível iterar. Se encontrar a solução pretendida ao longo das iterações a mesma será retornada. No caso de o limite fornecido não ser suficiente para encontrar a mesma retorna false.

Complexidade Temporal: $O(b^d)$

Complexidade Espacial: $O(bd)$

5.2 Procura informada

A procura informada utiliza o conhecimento específico do problema na escolha do próximo nodo a ser espandido. Esta introduz uma métrica que permite ao agente de busca estimar a distância desde o estado atual ao estado objetivo. Sendo assim mais eficiente do que a procura não informada.

Os vários tipos de procura informada abordados foram:

5.2.1 Gulosa

Este algortimo de procura resume-se à expansão do nó que parece estar mais perto da solução. Faz a análise dos nodos adjacentes ao nodo em que se encontra e dirige-se para o que tiver o custo aproximado, até ao destino, menor. No entanto, pode entrar em ciclos e é suscetível a falsos começos.

As maiores vantagens deste tipo de pesquisa são, nomeadamente, a sua simplicidade de fácil implementação e consequentemente a sua rápida execução de pesquisa. Em contrapartida, nem sempre conduz a uma solução ótima para o problema fornecido.

Este algoritmo é mais utilizado quando nos são mais fornecidas sequências de escolhas, optando em cada passo por escolher a solução que lhe parecer melhor, sem nunca reconsiderar essa mesma decisão. Nem sempre fornece a pesquisa mais ótima, embora muitas vezes o faça variando com a sequência de passos dados.

Para esta pesquisa, utiliza-se a estimativa de custo entre os dois nodos, que corresponde à distância em linha reta entre estes. Para isso, cada nodo guarda informação da distância em linha reta para os restantes nodos do grafo. Tornando assim esta procura de fácil implementação, sendo apenas necessário escolher o nodo com menor distância até se chegar ao nodo destino.

5.2.2 A* (A estrela)

Algoritmo A* é um algoritmo que procura um caminho num grafo de um vértice inicial até um vértice final. É a combinação de aproximações heurísticas como do algoritmo Breadth First Search (Busca em Largura) e da formalidade do Algoritmo de Dijkstra.

Este algoritmo evita expandir para caminhos de elevado custo optando pelo passo mais económico. É, então, um método de procura que busca otimizar a solução considerando sempre todas as informações disponiveis até esse instante, não apenas os da sua última expansão (ao contrário da pesquisa gulosa). Como dito anteriormente, ao combinar a pesquisa gulosa e com a uniforme, o algoritmo A* minimiza a soma do caminho já efetuado com o mínimo previsto que falta até á solução pretendida.

A* é mais utilizado em situações idênticas à do nosso projeto, ou seja, encontrar rotas de deslocamento entre vários pontos de uma cidade. Para além disso, é utilizada na resolução de problemas, quebra-cabeças e muito aproveitado em jogos.

Para esta pesquisa, utilizamos como heurística a soma do custo das arestas com a estimativa da distância do nodo que pretendemos ir, assim, esta pesquisa é mais completa que a gulosa pois tem em consideração o custo das arestas. E mantém em memória todas as opções em aberto até ter encontrado a solução. Esta é assim mais dispendiosa em termos de memória e tempo, mas é também mais completa do que as restantes pesquisas.

6 Análise de Resultados

Nas seguintes tabelas iremos analisar e comparar as diferentes pesquisas no nosso projeto. Para isso, decidimos escolher as seguintes localidades: Espinho, São Lázaro e Real. Optamos por três localidades a curta, média e longa distância, respetivamente. Esta seleção servirá para conseguir comparar os parâmetros (tempo, espaço, custo e melhor solução) das diferentes pesquisas.

Para uma melhor análise, os valores de tempo e de custo resultam de uma média de cinco testagens distintas.

Primeiramente, vamos testar os nossos algoritmos para circuitos entre Centro e Espinho.

Estratégia	Tempo (segundos)	Inferências	Indicador/custo (Km)	Melhor solução?
DFS	$4.08 * 10^{-5}$	204	45.3	Não
BFS	$6.34 * 10^{-5}$	212	6.1	Sim
Busca Iterativa	$2.71 * 10^{-5}$	28	6.1	Sim
Gulosa	$6.91 * 10^{-5}$	86	6.1	Sim
A*	$7.27 * 10^{-5}$	141	6.1	Sim

De seguida, efetuamos testes para circuitos entre Centro e São Lázaro.

Estratégia	Tempo (segundos)	Inferências	Indicador/custo (Km)	Melhor solução?
DFS	$4.91 * 10^{-5}$	114	34	Não
BFS	$9.42 * 10^{-5}$	245	6.6	Sim
Busca Iterativa	$4.14 * 10^{-5}$	46	6.6	Sim
Gulosa	$4.98 * 10^{-5}$	95	6.6	Sim
A*	0.0001	155	6.6	Sim

Por último, para circuitos entre Centro e Real.

Estratégia	Tempo (segundos)	Inferências	Indicador/custo (Km)	Melhor solução?
DFS	$2.75 * 10^{-5}$	63	22.6	Não
BFS	0.0017	1052	12.4	Sim
Busca Iterativa	$4.86 * 10^{-5}$	150	12.4	Sim
Gulosa	$6.70 * 10^{-5}$	182	12.4	Sim
A*	$6.70 * 10^{-5}$	182	12.4	Sim

Posto isto, chegamos à conclusão que:

- DFS : Nos três casos apresentou o maior custo, tempos relativamente baixos e nunca encontrou a melhor solução;

- BFS : Concluimos que o tempo de procura aumenta à medida que se aumenta a distância do destino, assim como o espaço ocuapdo em memória. Não é uma boa pesquisa para grande problemas. Quanto ao custo apresenta bons resultados, visto que encontrou sempre a melhor solução.

- Busca Iterativa : De forma geral, este algoritmo apresenta melhores tempos comparativamente aos restantes. Encontra sempre a melhor solução a curta, média e longa distância, logo o seu custo é igual aos das restantes pesquisas, exceto a DFS que apresenta sempre o maior.

- Gulosa : Em média, é a segunda pesquisa que mais tempo demora a encontrar a solução, mas encontra sempre a melhor solução. Como já foi dito anteriormente, apresenta um custo bastante positivo.

- A * : É a que apresenta maiores tempos de execução, porém é a pesquisa mais fiável e a que recorremos para resolver as queries fornecidas, visto que é a mais completa. Tal como foi dito anteriormente, devolve sempre a melhor solução e o têm um custo bastante psoitivo.

7 Conclusão e Comentários Finais

Inicialmente, surgiram algumas questões em relação à interpretação do enunciado, mas através da ajuda dos docentes e de alguns pontos de vista de colegas achamos ter chegado à solução mais alinhada com o enunciado proposto.

Assim, alcançámos todos os objetivos propostos e conseguimos, utilizando um cenário realista, verificar como as diferentes pesquisas atuam na nossa base de conhecimento para trajetos a curta, média e longa distância.

Desta forma, através da realização deste trabalho, obtivemos novos conhecimentos quanto às procuras (não informadas e informadas) ajudando assim a consolidar a matéria lecionada em aula pelos docentes.

Concluindo, enquanto grupo consideramos que dividimos bem o trabalho entre os quatro elementos e que, de forma geral, tivemos um aproveitamento positivo.