Pesquisa aplicada à entrega de encomendas

Relatório Final

Inteligência Artificial

3º ano do Mestrado Integrado em Engenharia Informática e Computação

Grupo A4_3

António Melo – up201403053 – up201403053@fe.up.pt

Bruno Santos – up201402962 – up201402962@fe.up.pt

21/05/2017

1. Objectivo

Este projecto foi desenvolvido no âmbito da unidade curricular de Inteligência Artificial e tem como objectivo determinar o percurso a realizar para efectuar a entrega de um conjunto de encomendas de uma empresa.

A cada encomenda está associada um valor, volume e ponto de entrega. O camião responsável por entregar as encomendas tem uma capacidade máxima e uma quantidade máxima de combustível que pode ter no depósito. Estas variáveis irão assim influenciar a escolha de que encomendas entregar (no caso de não ser possível entregar todas) e de que percurso fazer.

Para encontrar a solução para este problema são usados dois algoritmos de pesquisa de solução: Pesquisa em Largura e A*.

2. Especificação

2.1. Análise

No projecto realizado, o problema a resolver foi representado por um grafo pesado não direccionado, equivalente ao esquema de um mapa, com um vértice de partida e vários vértices de entrega e de reabastecimento.

O objectivo da solução encontrada é, começando no ao vértice de partida, passar pelos vários vértices de entrega e, no final, regressar ao vértice de partida.

São dois os cenários possíveis para o problema.

O primeiro será quando o combustível e a capacidade do camião são suficientes para acomodar a entrega de todas as encomendas. Neste caso apenas será necessário encontrar o circuito mais curto que ligue todos os pontos de entrega e o ponto de partida. Neste caso nunca será necessário reabastecer.

O outro será quando por causa da limitação de combustível, ou de carga, ou de ambos, não é possível entregar todas as encomendas. Neste caso será necessário escolher as encomendas a entregar. Para isso, o programa permite ao utilizador especificar se pretende maximizar o número de entregas feitas ou valor das mesmas. Se a limitação existente for devido ao combustível necessário, o utilizador pode escolher se pretende reabastecer, ou não, ao longo do percurso.

2.2. Abordagem

Através da análise do problema pudemos concluir que o problema se resume ao problema do caixeiro-viajante.

Considerou-se então que os vértices do problema são os vértices de partida e entrega (e de reabastecimento se for o caso) e as arestas são os caminhos mais curtos entre cada par destes vértices (obtido através do algoritmo A* com heurística de distância linear ao objectivo).

Desta forma, para os algoritmos de procura de solução usados, os estados são caminhos possíveis que nunca passam pelo mesmo vértice de entrega mais do que uma vez. O número de resultados possíveis para entregas completas é o número de permutações dos vértices do problema (n!) portanto o problema cresce bastante rápido em ordem ao número de entregas a fazer.

Como referido anteriormente, foram usados dois algoritmos de pesquisa de soluções: Pesquisa em Largura e A*.

2.2.1. Pesquisa em Largura

O algoritmo de pesquisa em largura permite percorrer todos os vértices de um grafo. Partindo da raiz, explora todos os vértices vizinhos até passar para o próximo nível.

Neste problema o algoritmo começa com o caminho unitário [1], prosseguindo para os caminhos [1,2], [1,3], …, [1,n], e de seguida [1,2,1], [1,2,3], …, [1,2,n], [1,3,1], [1,3,2], … e assim sucessivamente, garantindo que todos os caminhos não têm vértices repetidos (excepto o vértice de partida).

Há medida que o algoritmo percorre os estados possíveis, é guardado o melhor circuito (percurso que começa e termina no vértice de partida). A cada momento o melhor circuito será aquele que tiver o maior número de vértices ou maior valor total de encomendas entregues (dependendo do escolhido pelo utilizador). Se dois percursos tiverem as variáveis anteriores iguais o melhor será aquela que percorre uma distância menor.

Para reduzir o número de estados a pesquisar não são explorados os estados (ignora-se os filhos destes), cujos custos (combustível ou carga) são superiores aos limites estabelecidos.

Quando o utilizador pretende que o camião abasteça no seu percurso, a passagem por estes vértices também é tida em conta como se se tratasse de uma passagem por um vértice de entrega sendo, no

entanto, o combustível do camião retornado ao seu limite máximo.

Devido ao crescimento acelerado da complexidade do problema, esta solução rapidamente se torna pouco viável na pesquisa da solução óptima de cada cenário.

2.2.2. Algoritmo A*

O algoritmo A* é um algoritmo de pesquisa informada que mistura o algoritmo de custo uniforme com métodos gananciosos. Avalia cada estado combinando g(n), o custo para atingir esse estado, com h(n), uma heurística que estima o custo do caminho menor para chegar ao objectivo a partir desse estado.

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

A cada iteração escolhe e expande o estado cujo f(n) seja menor até encontrar o estado final.

Para o algoritmo encontrar o caminho mais curto, a função heurística usada tem que ser admissível, ou seja, não pode sobrestimar o custo real para chegar ao objectivo.

Para se implementar o algoritmo A* procedeu-se da mesma forma que com a pesquisa em largura: cada estado representa um caminho possível entre os vértices a ponderar.

A função de custo usada foi:

$$\max\left(\frac{distância * combustível/km}{combustível\ total}, \frac{carga}{carga\ total}\right)$$

que estabelece que, para cada estado, o custo para o alcançar será o maior uso percentual de recursos para necessários para tal.

A função heurística, para além de ter de ser admissível, tem também de ser rápida o suficiente de calcular de modo a não atrasar significativamente o algoritmo. Desta forma como função heurística foi usada a menor distância entre o último vértice do percurso e o vértice de partida/chegada, que cumpre os dois requisitos anteriores.

Neste algoritmo, tal como na pesquisa em largura o melhor circuito é guardado à medida que os estados são percorridos.

O algoritmo pára quando é encontrado um percurso que passe por todos os vértices de entrega ou quando o percurso a ser expandido ultrapassa as restrições estabelecidas. Neste último caso, é retornado o melhor circuito encontrado até ao momento.

3. Desenvolvimento

3.1. Ferramentas de Desenvolvimento

Para o desenvolvimento do projecto usámos a linguagem de programação Java no IDE Eclipse Neon.

Quanto a *APIs* fizemos recurso à biblioteca *GraphStream* para a visualização dos resultados e à *Java API for XML Processing (JAXP)*, para criar e ler ficheiros *XML* que representam os dados de cada problema.

3.2. Estrutura

O programa foi dividido em quatro módulos:

- userInterface – para a interface com o utilizador;
- xmlParser – para criar e ler os ficheiros *XML* que representam os dados de cada problema;
- problemData – para representar os dados dos problemas;
- algorithms – para resolver os problemas.

3.3. Detalhes relevantes

Como referido anteriormente, é possível criar e ler ficheiros *XML* com dados de problemas. A criação de problemas aleatórios é também possível através da interface com o utilizador, sendo necessário que este introduza valores de referência (número de nós, número de encomendas a entregar…).

Foram feitas algumas optimizações ao nível de estruturas internas, como usar *priority queues* para o algoritmo A* e guardar as distâncias calculadas entre nós, para reduzir o tempo de execução.

4. Experiências

4.1. Objectivos

Foram criados vários cenários de diferentes dimensões de modo a comparar a qualidade e os recursos utilizados para obter soluções com ambos os algoritmos implementados.

Como a pesquisa em largura retorna sempre a melhor solução para o problema, isto também permitiu validar os resultados obtidos através do algoritmo A*.

4.2. Resultados

Foram testados vários cenários com combinações diferentes de dimensão, restrições (carga/ combustível/ ambas/ nenhuma), optimizações (distância/ valor) e reabastecimento (sim/ não).

Quase todos estes cenários obtiveram o resultado óptimo no algoritmo A* (comprovado pela pesquisa em largura), exceptuando alguns casos em que é necessário o reabastecimento do camião.

De seguida apresentam-se as comparações dos recursos utilizados por cada algoritmo, em função da complexidade do problema, para encontrar o caminho mais curto para entregar todas as encomendas, sem restrições de combustível ou carga.

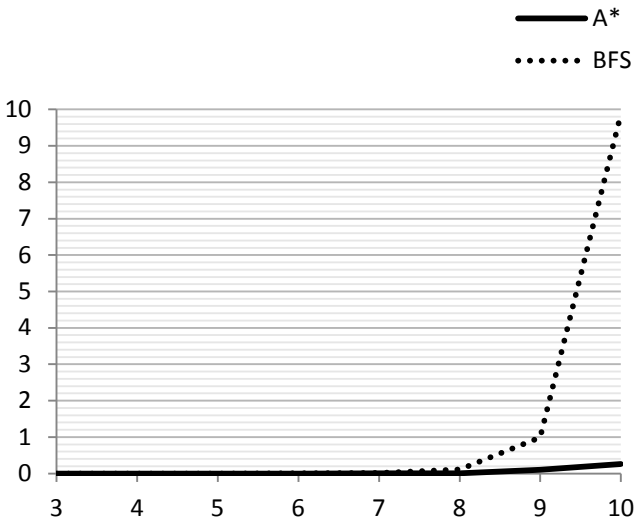

Fig. 1 – Número de percursos explorados (milhões) em funções do número de entregas a fazer

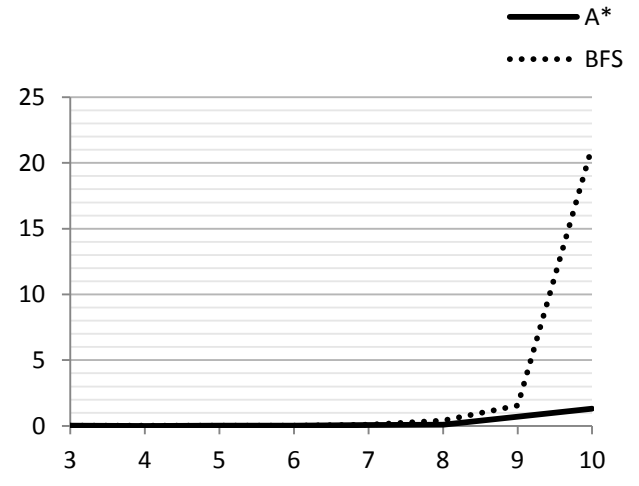

Fig. 2 – Tempo de execução (segundos) em funções do número de entregas a fazer

5. Conclusões

Analisando-se os resultados obtidos, verifica-se que tanto o algoritmo A* como a pesquisa em largura obtêm resultados idênticos, sendo que o A* consegue geralmente fazê-lo em menos tempo e como menos iterações, sendo esta diferença tanto mais evidente quanto maior for o problema.

Como referido anteriormente, quase todos os cenários foram testados com sucesso exceptuando alguns que implicavam o reabastecimento do camião. Esta inconsistência provavelmente deve-se ao facto de ser necessário usar uma função de custo e heurística diferentes considerando estes vértices como "especiais" para obter melhores resultados.

Apesar destes problemas, em geral os resultados vão de encontro ao esperado e permitem comprovar a validade da função de custo e heurística usada para o algoritmo A* para os casos em que não é necessário reabastecer.

6. Melhoramentos

No futuro seria interessante melhorar a função de custo e heurística do algoritmo A* para lidar com os problemas referidos anteriormente, assim como implementar mais algoritmos de pesquisa sistemática/informada de soluções para melhor se poder comparar resultados.

7. Recursos

7.1. Bibliografia

Russel, Stuart; Norvig, Peter – Artificial Intelligence: A Modern Approach, 3rd Edition.

Oliveira, Eugénio – Métodos de Resolução de Problemas e Algoritmos para a Evolução, Inteligência Artificial, MIEIC FEUP, https://paginas.fe.up.pt/~eol/IA/1617/ia_.html

7.2. Software

Eclipse Neon

7.3. Elementos do grupo

António Melo – up201403053 – 50%

Bruno Santos – up201402962 – 50%

8. Apêndice – Manual de Utilizador

O *main* do projecto encontra-se na classe *UserInterface* no *package userInterface*.

Ao executar são apresentadas as opções de criar um novo ficheiro ou ler um já existente.

Na opção de criar um novo ficheiro é possível introduzir os vários parâmetros de configuração de um problema. É relativamente fácil de inferir o que cada parâmetro significa à excepção de *Connection level* que será explicado seguidamente. No processo de criação é garantido que todos os vértices gerados estão conectados aos restantes. O valor de *Connection level* (inteiro maior ou igual a zero) aumenta o número de conexões adicionais feitas pelo gerador em *nrNodes*connectionLevel*.

No final da geração do grafo do problema este é apresentado usado o GraphStream e o ficheiro de configuração é guardado no directório *data* contida no projecto.

Ao se escolher ler um ficheiro existente, é possível obter o melhor caminho com base nos dados lidos.

Em "Document name: ", o utilizador deve introduzir o *path* a partir no directório *data* e o nome do ficheiro, sem a extensão. Pode depois escolher que parâmetro optimizar (número de entregas ou valor total das mesmas) e se pretende reabastecer ao longo do caminho ou não.

Após isto, são apresentados dois grafos: um com um percurso a verde que representa o resultado obtido com o algoritmo A* e um a amarelo com o resultado da pesquisa em largura, sendo possível visualizar a evolução do melhor percurso encontrado até ao momento em ambos.