­

Autores:

Gonçalo Monteiro – 1190624

Luís Costa – 1190825

Lara Domingos – 1190797

Luís Carlos - 1190818

Relatório ALGAV

2022

Redigido: 03/12/2022

Sprint B

## Introdução

No âmbito da unidade curricular Laboratório / Projeto V, lecionado na Instituto Politécnico do Porto, foi requisitado o desenvolvimento de múltiplos algoritmos necessários para o funcionamento de uma aplicação web denominada de EletricGo.

No seguinte relatório apresentamos ao leitor os algoritmos desenvolvidos, juntamente com uma explicação sucinta dos objetivos pretendido, seguido por fim de uma análise da complexidade temporal de cada algoritmo.

Para simplificar a análise da complexidade foi adotada a notação Big-O. Esta notação representa o limite superior do tempo de execução de um qualquer algoritmo, logo fornece a complexidade de um dado algoritmo correspondente ao pior cenário. A adoção de uma anotação tem o benefício acrescido de medir a eficiência dos algoritmos sem ter em consideração especificidades de hardware. Como interesse académico iremos, contudo, apresentar ao leitor dados estatísticos do tempo de execução dos algoritmos consoante o tamanho do input.

## Brute-Force: Geração de todas as Rotas

A aplicação web desenvolvida dedica-se principalmente à gestão de uma frota de camiões e de entregas, para tal é de alta importância a que aplicação disponibilize um serviço que permita calcular todas as rotas possíveis para uma dada lista de entregas. O algoritmo que se segue permite então ao sistema EletricGo responder à necessidade previamente descrita.

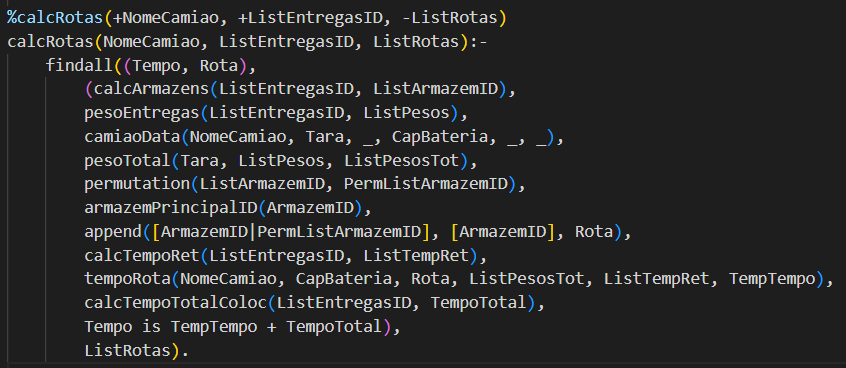


Figura 1 - Predicado que calcula todas as rotas possíveis para a realização de uma dada lista de entregas;

Ao analisarmos o predicado “calcRotas”, constatamos que invoca o predicado “findall”, que por sua vez, no contexto em causa, produz uma lista que contem pares (cada um composto por um elemento e uma lista) que respeitam os objetivos traçados, isto é o predicado “findall” retorna uma lista que contem os tempos associados a cada rota possível para uma dada lista de encomendas.

Constatamos que os objetivos definidos no predicado “findall” têm maioritariamente uma complexidade temporal O(n), pois cada um deles limita-se a percorrer uma lista passado como argumento do predicado. Por questões de simplificações iremos simplesmente analisar o predicado “tempoRota”, contudo é possível consultar os restantes predicados no repositório público do projeto ([S5\_DA\_02/Planeamento.pl at main · 1190624/S5\_DA\_02 (github.com)](https://github.com/1190624/S5_DA_02/blob/main/Planeamento/Planeamento.pl)).

Uma imagem com texto

Descrição gerada automaticamente

Figura 2 - Predicado que calcula o tempo associado a uma dada rota;

Apesar de existir várias alternativas em caso de insucesso, com a exceção do caso base do predicado, todos eles possuem uma complexidade O(n), em que “n” corresponde ao número de elementos das listas “ListPesosTot” e “ListEntregasID” e ao número de pares de elementos únicos consecutivos da lista “Rota”. É possível chegar a tal conclusão pois, com a exceção do caso base, cada uma das alternativas efetua unicamente operações primitivas (estas possuem uma complexidade temporal O(1)) e efetuam uma chamada recursiva que conclui unicamente as listas acima mencionadas se encontram vazias (no caso da lista “Rota”, quando esta possui unicamente 1 elemento).

Logo recorrendo a regras de simplificação é possível concluir que a complexidade do predicado “calcRotas” é O(n! . n), pois visto que o predicado determina todas as rotas possíveis associadas a uma dada lista de entregas, é necessário efetuar uma permutação de todos os armazéns que serão visitados, o que resulta por si só numa complexidade temporal de “n!”, onde “n” representa o número de armazéns a visitar para uma dada lista de entregas.

## Rota mais rapida dado um camião e lista de entregas

O predicado desenvolvido procura encontrar a rota possível com menos tempo consumido para todo o processo de entrega e regresso a origem. Com o uso do predicado “calcRotas”, é necessário apenas realizar um sort para determinar a rota com o menor tempo, dentro das rotas possíveis.

Uma imagem com texto

Descrição gerada automaticamente

Figura 3 - Predicado que determina a rota mais rápida;

O resultado do sort é guardado numa lista temporária e por fim é retirada a rota mais rápida através da cabeça da resultante lista temporária.



Figura 4 – Apresentação do resultado do predicado da Figura 3;

Quanto a complexidade podemos considerar que é um resultado das complexidades do predicado “calcRotas” apresentada acima e do sort. Visto que as listas em em prolog são imutáveis, não pode ser aplicado diretamente o O(nlog n) de um sort dito comum. Tendo em conta a pesquisa realizada o a complexidade do sort pode tanto pode ser O(nlog n) como no pior dos casos O(n). Sendo assim no pior dos casos o predicado apresentará uma complexidade resultante de O(n! . 2n), simplificando O(n! . n).

## Calcular o tempo de execução da rota mais rápida

Como forma de compreender até que ponto é afetado o tempo de execução do predicado anterior ao serem adicionadas cada vez mais entregas a serem em entregues a diferentes armazéns.

Uma imagem com texto

Descrição gerada automaticamente

Figura 5 –Predicado que calcula o tempo de execução do predicado da Figura3;

Através de chamadas consecutivas do predicado, alterando apenas o número de encomendas, podemos verificar que o tempo de execução continua a aumentar exponencialmente, previsto tendo em conta a complexidade calculada no ponto anterior.

Uma imagem com texto

Descrição gerada automaticamente

Figura 6 – Apresentação do resultado do predicado da Figura 5;

Sendo que a escala até ao momento ainda é pequena o tempo de execução gasto é desprezável, mas num ambiente em que muitos mais armazéns são adicionados a rede, e tendo em conta que uma complexidade temporal de O(n! . n), recorrer a um gerador de todas as soluções acaba por se tornar dispendioso temporalmente.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nº de armazéns  (neste estudo cada entrega usada tem 1 armazém diferente) | Lista com a sequência de armazéns para as entregas | Ordem Armazéns a visitar | Tempo para fazer as entregas | Tempo de geração da solução (TSol) |
| 1 | [4439] | [5,1,5] | 255.35 | 4.29e-5 |
| 2 | [4439,4438] | [5,9,1,5] | 392.3 | 8.20e-5 |
| 3 | [4439,4438,4445] | [5,9,3,1,5] | 380.44 | 1.56e-4 |
| 4 | [4439,4438,4445,4443] | [5,8,1,3,9,5] | 392.67 | 1.07e-3 |
| 5 | [4439,4438,4445,4443,4449] | [5,8,1,3,11,9,5] | 513.11 | 3.22e-3 |
| 6 | [4439,4438,4445,4443,4449,4450] | [5,8,17,1,3,11,9,5] | 557.84 | 2.35e-2 |

Para melhorar a tabela cima foi também criado o seguinte predicado como forma a representar o tempo para fazer as entregas.

Uma imagem com texto

Descrição gerada automaticamente

Figura 7 – Predicado da Figura 3 alterado para apresentar também o tempo da entrega;

## Heurísticas

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nº de armazéns | Armazéns a percorrer | Tempo para Entregas Heurística do menor tempo ou distância | Tempo para Entregas Heurística da maior massa | Tempo para Entregas Heurística combinada | Melhor solução pelas 3 heurísticas |
| 4 | [1,2,6,4] | 5.38e-5 | 3.81e-5 | 6.48e-5 | Massa |
| 5 | [1,2,6,4,7] | 1.02e-4 | 4.88e-5 | 6.60e-5 | Massa |
| 6 | [1,2,6,4,7,8] | 7.20e-5 | 8.58e-5 | 5.41e-5 | Combinada |
| 7 | [1,2,6,4,7,8,9] | 9.98e-5 | 5.50e-5 | 1.06e-5 | Combinada |
| 8 | [1,2,6,4,7,8,9,10] | 1.05e-5 | 7.60e-5 | 1.07e-4 | Distância |
| 9 | [1,2,6,4,7,8,9,10,3] | 1.14e-4 | 1.36e-5 | 1.43e-4 | Massa |
| 10 | [1,2,6,4,7,8,9,10,3,11] | 1.30e-4 | 9.51e-5 | 1.56e-4 | Massa |
| 11 | [1,2,6,4,7,8,9,10,3,11,12] | 1.33e-4 | 1.01e-4 | 1.80e-4 | Massa |
| 12 | [1,2,6,4,7,8,9,10,3,11,12,13] | 1.50e-4 | 1.15e-4 | 2.14e-4 | Massa |

|  |  |  |
| --- | --- | --- |
| Media - Tempo para Entregas Heurística do menor tempo ou distância | Média - Tempo para Entregas Heurística da maior massa | Média - Tempo para Entregas Heurística combinada |
| 0.961e-4 | **0.698e-4** | **1.10e-4** |