Análise e Síntese de Algoritmos 2015/2016

2º Projeto - Relatório

Introdução

A tarefa deste projeto consiste em encontrar um ponto de encontro adequado que permita juntar os empregados de todas as filiais da empresa Coelho e Caracol Lda.

Para minimizar os custos das rotas de cada filial até ao ponto de encontro, os empregados podem fazer entregas ao longo do percurso. Uma rota consiste numa sequência de localidades, sendo que a cada par de localidades está associado um valor de perda que resulta de subtrair a receita ao custo.

Cada filial estabelece para cada localidade qual é o percurso que, com origem nessa filial, minimiza o valor de perda total . Assim, o ponto de encontro irá corresponder à localidade em que a perda total da empresa, tendo conta todas as filiais, é minimizada.

Sabemos ainda que: o número de filiais é significativamente menor que o número de localidades; o número de pares de localidades entre os quais há um valor de perda é muito menor que o número de todos os pares possíveis; e não existe um percurso entre localidades que forme um ciclo com valor de perda total negativo.

Descrição da solução

Com base nos dados do problema, podemos começar por representar as localidades e rotas associadas através de um grafo dirigido pesado, visto que a cada par de localidades está associado um valor de perda.

Neste contexto, podemos verificar que a solução do problema consiste em determinar os caminhos mais curtos de cada filial a todas as outras localidades, isto é os caminhos que minimizam o valor de perda. Com base nestes caminhos, para obter o ponto de encontro precisamos apenas de identificar a localidade que, considerando todas as filiais, minimiza a perda total da empresa.

Considerando a representação escolhida e as restantes limitações para o problema, podemos recorrer a um algoritmo que calcule os caminhos mais curtos entre as localidades que têm filiais e todos as localidades da rede. Tendo em conta as limitações do problema,

sabemos que o grafo vai ser esparso e não pode ter ciclos negativos, porque o número de pares de localidades entre os quais há um valor de perda é muito menor que o número de todos os pares possíveis e não existe um percurso entre localidades que forme um ciclo com valor de perda total negativo.

Como o número de filiais é significativamente menor que o número de localidades, um algoritmo que calcule os caminhos mais curtos entre todos os pares de vértices do grafo (por exemplo, Johnson ou Floyd-Warshall) é menos eficiente do que uma solução que utilize apenas o algoritmo de Bellman-Ford.

Seja V o número de vértices (localidades), E o número de arcos (caminhos entre pares de localidades) e F o número de filiais, a solução apresenta a seguinte estrutura:

1. Simula o algoritmo de Bellman-Ford no grafo - complexidade O(FVE) - e guarda os valores da perda total minimizada em cada localidade, considerando cada uma das filiais
2. Identifica o ponto de encontro comparando os valores de perda total à procura do valor mais baixo.
3. Simula o algoritmo de Bellman-Ford novamente – complexidade O(FVE) - e, desta vez, guarda o valor de perda no ponto de encontro em relação a cada filial.

Análise Teórica

A análise teórica da solução consiste em descobrir a sua complexidade com base na complexidade das suas componentes. Nesta análise, o número de vértices do grafo é representado pela letra V, o número de arcos pela letra E e o número de filiais pela letra F.

A leitura da primeira linha de input (número de vértices, filiais e arcos) ocorre em tempo de constante – O(1).

A leitura da segunda linha de input corresponde a um ciclo que lê do input as localidades que são filiais com tempo de execução O(F).

A inicialização do grafo corresponde a um ciclo que lê do input os arcos do grafo e os adiciona ao vector de arcos em tempo constante O(1). O tempo de execução deste ciclo depende do número de arcos, ou seja, é O(E).

A primeira aplicação do Bellman-Ford ocorre com complexidade temporal O(FVE), porque cada iteração do algoritmo (O(VE)) é feita F vezes. A soma dos valores da perda total associada a cada filial com todas as localidades é um ciclo com tempo de execução O(V).

Identificar o ponto de encontro com base nos valores de perda através da função get_ponto_encontro tem complexidade linear devido à complexidade temporal associada ao método min_element.

A segunda aplicação do Bellman-Ford, utilizada para obter a perda associada a cada filial com o ponto de encontro identificado, tem também complexidade temporal O(FVE).

Finalmente, a apresentação do ponto de encontro, perda total da empresa e perda associada a cada filial para esse ponto de encontro tem complexidade temporal O(F).

Analisando todas as componentes da solução, podemos concluir que a sua complexidade teórica é O(FVE).

Avaliação Experimental

Na avaliação experimental da solução, foram utilizados os seis testes públicos fornecidos na página da cadeira para este projeto e cinco testes com variação do número de filiais e registados os seguintes resultados.

Vértices	Arestas	Filiais	Tempo (s)
5	6	3	0,004
2	2	2	0,004
100	800	10	0,007
100	1000	15	0,010
100	1000	10	0,008
250	5000	10	0,038
300	5000	2	0,014
300	5000	5	0,025
300	5000	10	0,042
300	5000	15	0,054

300	5000	20	0,070

Recorrendo à análise da tabela de resultados, podemos verificar que o tempo de execução tem um crescimento linearmente proporcional com o número de vértices, número de arestas e número de filiais, o que já tínhamos previsto com base na análise teórica da complexidade da solução - O(FVE).

Referências

https://en.wikipedia.org/wiki/Johnson%27s_algorithm

https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

https://en.wikipedia.org/wiki/Bellman%E2%80%93Ford_algorithm

https://en.wikipedia.org/wiki/Floyd%E2%80%93Warshall_algorithm

https://gist.github.com/ashleyholman/6793360

http://www.geeksforgeeks.org/greedy-algorithms-set-7-dijkstras-algorithm-for-adjacency-list-representation/