K-shortest path

Ingrid Santana Lopes
14/0083065
Departamento de Ciência da Computação,
Universidade de Brasília
Brasília,Brasil

Marcos Paulo Cayres Rosa
14/0027131
Departamento de Ciência da Computação,
Universidade de Brasília
Brasília,Brasil

I. K-SHORTEST PATH

K shortest path corresponde ao processo de se encontrar os k caminhos mais curtos entre um par de nodes em um grafo [2]. Para tal problema, diversos métodos foram propostos, analisados e comparados na literatura levando a algumas das análises aqui feitas e usadas de base para a implementação do algoritmo de k-shortest path.

Os métodos usados para se calcular os caminhos incluem diversos problemas e detalhes que devem ser levados em conta na implementação. Muitos problemas de otimização, por exemplo, podem ser resolvidos por programação dinâmica ou técnicas de busca em matrizes mais eficientes[2].

A seguir alguns métodos de implementação do k-shortest path encontrados na literatura são analisados para posteriormente o modelo usado ser apresentado.

A. Alguns Métodos

1) Algoritmo de Yen: é um algoritmo clássico para resolver o problema de achar k caminhos da origem para um destino [4] que tenham os menores comprimentos de forma que não são permitidos quaisquer loops ou pesos negativos [5]. O algoritmo pode ser dividido em duas partes, determinar o primeiro k-shortest path e então determinar todos os demais k-shortest path através de um método que usa uma váriável denominada A que será responsável por armazenar o k-shortest path enquanto B será responsável por armazer caminhos em potencial [5].

Por ser um algoritmo clássico, ele está presente em estudos [4, 3] que sugerem novas implementações, abordagens ou fazem estudos quanto a ele.

2) Algoritmo de David Eppstein: tem como ideia principal encontrar os caminhos de uma origem s para um destino t através de simplesmente encontrar caminhos saídos de s par aqualquer outro nó e o concatenando à um caminho mais curto a esse nó para t [2].

Uma parte de extrema importância do método é a construção de uma head binária para cada vértice listando também arestas que não são parte do caminho de menor custo [2].

3) Algoritmo de Dijkstra: é, teoricamente, o algoritmo mais eficiente para resolver o problema do k-shortest path [1], tal algoritmo mantém um custo experimental para cada vértice v e, conforme o algoritmo é executado, tal custo experimental vai decrescendo até atingir o custo mínimo final [1].

O algoritmo mantem uma divisória de vértices em três estados:

- *vértices não marcados* cujo custo é infinito [1].
- *vértices marcados* de custo finito mas cujo falor mínimo ainda não foi encontrado [1].
- *vértices scaneados* cujo custo mínimo já foi encontrado [1].

B. Método Utilizado

O método utilizado para a implementação do k-shortest path usa como base a rede de educação e pesquisa dos Estados Unidos, o National Science Foundation Network (NSFNET) como mostrado na figura 1:

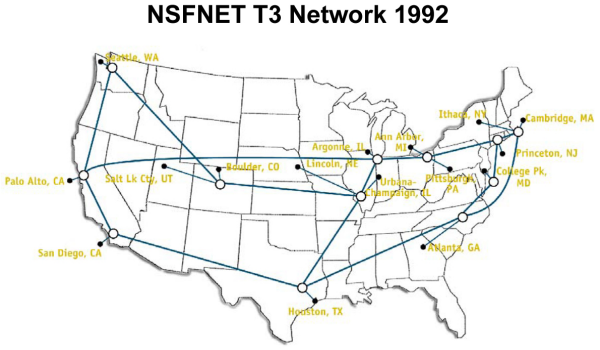

Figura 1. Sistema do NSFNET

Com base nas cidades visualizadas no esquema da figura 1, para facilitar nos cálculos e para quesitos de organização, a cada uma foi dado um número como mostrado na lista abaixo (com os pesos conforme encontrado em imagens pesquisadas):

0) Seattle
1) Palo Alto
2) San Diego
3) Salt Lake City
4) Boulder
5) Houston
6) Lincoln
7) Champaign
8) Atlanta
9) Pittsburgh
10) Ann Arbor
11) Ithaca

12) Princeton
13) College PK

Observa-se que, caso desejado, o código permite a leitura de um arquivo texto com os dados de uma nova rede ou a alteração no próprio código da matriz de pesos.

Com base nos métodos estudados e na estrutura criada com base na rede do NSFNET, foi utilizado o Algoritmo de Dijkstra já mencionado anteriormente. Entretanto, em sua forma original, o algoritmo só é capaz de achar o menor caminho entre dois nós, um de origem e um de destino, dessa forma, foi necessário fazer alterações para que ele pudesse achar os demais k-shortest paths. Para isso, foi decidido que a última aresta usada para chegar até o menor dos caminhos seria desconsiderada para o próximo cálculo e assim em diante até que ele não encontra mais k-shortest path.

A deleção da aresta foi baseada a partir de algoritmos que sugeriam esse método. No código implementado isso é feito ao final de cada chamada do Dijkstra, colocando os pesos como infinito e utilizando o vetor de nós anteriores integrante do algoritmo de Dijkstra. Ademais, o Dijkstra será chamado pelas k vezes definidas e executará até que chegue ao valor de k ou que não haja mais caminhos possíveis para o algoritmo.

Referências

[1] Ravindra K Ahuja et al. "Faster algorithms for the shortest path problem". Em: *Journal of the ACM (JACM)* 37.2 (1990), pp. 213–223.

[2] David Eppstein. "Finding the k shortest paths". Em: *SIAM Journal on computing* 28.2 (1998), pp. 652–673.

[3] Gabriel Y Handler e Israel Zang. "A dual algorithm for the constrained shortest path problem". Em: *Networks* 10.4 (1980), pp. 293–309.

[4] Ernesto QV Martins e Marta MB Pascoal. "A new implementation of Yen's ranking loopless paths algorithm". Em: *4OR: A Quarterly Journal of Operations Research* 1.2 (2003), pp. 121–133.

[5] Jin Y. Yen. "Finding the K Shortest Loopless Paths in a Network". Em: *management Science* 17.11 (1971), pp. 712–716.