OTIMIZAÇÃO DE REDES

Os problemas de otimização de redes podem ocorrer em várias áreas, mas geralmente são encontrados nas áreas de transportes e comunicações. Um problema típico de transporte consiste em encontrar uma rota, partindo de uma origem para um destino, considerando que entre esses pontos existem diversas rotas alternativas e que necessita-se minimizar ou maximizar alguma medida associada aos arcos e/ou nós. Existem outros problemas em que se necessita minimizar os valores associados aos arcos, de forma que possa atender todos os pontos de uma rede. A seguir serão relacionados vários algoritmos que objetivam a modelagem de redes.

1. PROBLEMAS DE MINIMIZAÇÃO DE REDES

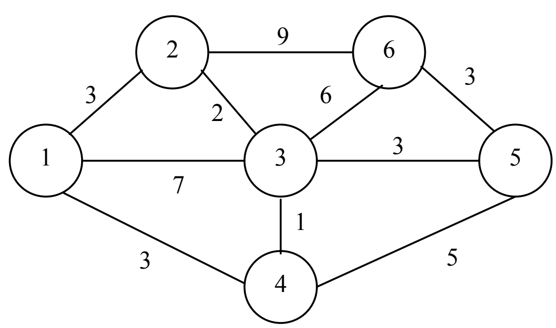
Os algoritmos de minimização de redes tratam da árvore de valor mínimo em problemas de interligação de redes não orientadas de comunicação, luz, água, esgoto, minerodutos, gasodutos etc. com o objetivo de atender todos os nós de uma rede, minimizando o consumo dos meios.

1.1. Algoritmo de PRIM

Este algoritmo compreende os seguintes passos:

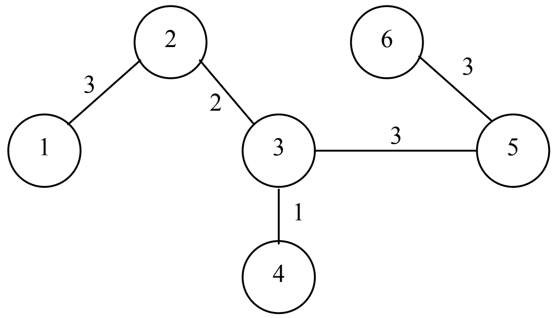
1º passo: selecionar qualquer nó da rede e o inserir no conjunto C (árvore mínima). O conjunto C* é formado pelos nós não conectados.
2º passo: identificar o nó do conjunto C* que está mais próximo de qualquer um dos nós do conjunto C. Deve-se repetir este processo até que todos os nós estejam conectados (C* = ∅).

Exemplo: Considere o grafo a seguir e avalie quais ligações que deverão ser implantadas visando a interligação de todos os nós, porém, considerando uma quilometragem total mínima. Os atributos dos arcos representam as distâncias entre as regiões.



C1 = { 4 } e C*1 = { 1,2,3,5,6 } => C2 = { 4,3 } e C*2 = { 1,2,5,6 } =>
C3 = { 4,3,2 } e C*3 = { 1,5,6 } => C4 = { 4,3,2,1 } e C*4 = { 5,6 } =>
C5 = { 4,3,2,1,5 } e C*5 = { 6 } => C6 = { 4,3,2,1,5,6 } e C*6 = ∅

Resultado Final: 12Km



1.2.Algoritmo de Kruskal

Deve-se construir uma árvore, selecionando-se arcos, iniciando-se pelo arco de menor atributo, adicionando-os em ordem crescente de atributos, de modo a não formar ciclos com os arcos já selecionados. O "ponto de parada" do algoritmo é identificado quando a árvore possuir n-1 arcos conectados, sendo "n" o número de nós do grafo.

Este algoritmo compreende os seguintes passos:

1º passo: colocar os arcos em ordem crescente de valor. Estes arcos fazem parte de um conjunto "A*" de arcos não conectados. Inicialmente A é vazio, ou seja, A=∅.
2º passo: selecionar o menor dos arcos de A* que não forme um ciclo com os demais e coloque-o no conjunto A. Um arco forma um ciclo quando os vértices deste arco já fazem parte da árvore mínima em construção.
3º passo: se A possui n-1 arcos, sendo "n" o número de nós, deve-se parar o algoritmo, pois os arcos de A compõem a árvore mínima. Caso contrário voltar para o passo 2.

Exemplo: Utilizando o mesmo grafo do exemplo anterior, identifique a árvore mínima pelo algoritmo de Kruskal.

Passo 1 A* = {(3,4),(3,2),(1,2),(3,5),(6,5),(1,4),(4,5),(3,6),(3,1),(2,6)}
A = ∅
Passo 2 A = {(3,4)}
A* = {(3,2),(1,2),(3,5),(6,5),(1,4),(4,5),(3,6),(3,1),(2,6)}
Passo 3 n = 6 e n-1 = 5
O número de elementos de A é igual a 1, e como n(A) < 5, deve-se retornar ao passo 2.

Passo 2 A = {(3,4), (3,2)}
A* = {(1,2),(3,5),(6,5),(1,4),(4,5),(3,6),(3,1),(2,6)}
Passo 3 n = 6 e n-1 = 5
O número de elementos de A é igual a 2, e como n(A) < 5, deve-se retornar ao passo 2.

Passo 2 A = {(3,4), (3,2), (1,2)}
A* = {(3,5),(6,5),(1,4),(4,5),(3,6),(3,1),(2,6)}

Passo 3 n = 6 e n-1 = 5
O número de elementos de A é igual a 3, e como n(A) < 5, deve-se retornar ao passo 2.

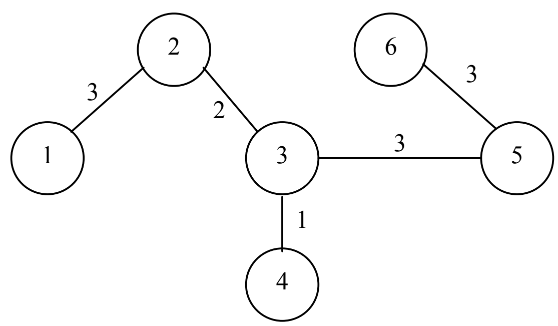
Passo 2 A = {(3,4), (3,2), (1,2), (3,5)}
A* = {(6,5),(1,4),(4,5),(3,6),(3,1),(2,6)}

Passo 3 n = 6 e n-1 = 5
O número de elementos de A é igual a 4, e como n(A) < 5, deve-se retornar ao passo 2.

Passo 2 A = {(3,4), (3,2), (1,2), (3,5), (6,5)}
A* = {(1,4),(4,5),(3,6),(3,1),(2,6)}

Passo 3 n = 6 e n-1 = 5
O número de elementos de A é igual a 5, e como n(A) = 5, deve-se parar o processo de análise.

Resultado Final: 12Km



2. CAMINHO MÍNIMO

Em uma rede, dependendo das suas características construtivas, podem existir vários caminhos entre um par de nós (origem/destino). Entre os caminhos possíveis, aquele que possui menor "peso" é chamado de caminho mínimo. Este peso pode ser representado pela soma dos atributos dos arcos que formam o caminho, tais como, tempo de viagem, distância percorrida etc..

Para resolver problemas desse tipo, há vários algoritmos (Ford, Faude, Bellman, Dijkstra, Floyd, Hasse dentre outros) que envolvem maior ou menor complexidade de cálculo (número de operações elementares, tais como adição, subtração, multiplicação etc.).

2.1. Algoritmo de Dijkstra

Este algoritmo foi desenvolvido em 1959 e posteriormente Dantizg (1960) e Nicholson (1960) desenvolveram um algoritmo de duas árvores de Dijkstra, cuja idéia é construir árvores de caminhos mínimos de um nó de origem e de um nó de destino, simultaneamente.

O Algoritmo de Dijkstra é utilizado para determinar o caminho mínimo de um nó para outro nó ou para todos os outros nós da rede, considerando que os atributos dos arcos são positivos. Todos os arcos devem ser orientados.

Nele, considera-se que um nó é "fechado" quando se encontra o caminho mínimo da origem até este nó, e aqueles nós cujos caminhos mínimos ainda não foram encontrados são considerados "abertos".

O conceito de fechado ou aberto está associado à impossibilidade de encontrar um caminho melhor do que o já encontrado, assim enquanto o nó não é fechado (ou rotulado) ainda é possível encontrar um caminho de menor valor até este nó.

Este algoritmo compreende os seguintes passos:

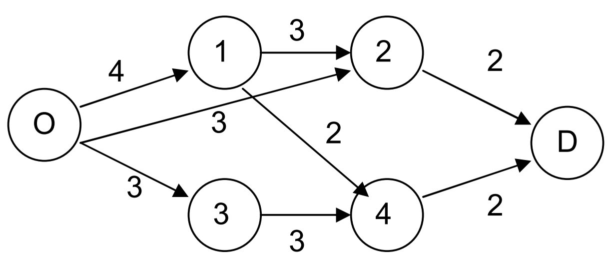
1º passo: considerando que $d(x)^{i} = \min\{d(x)^{i-1}, d(y) + d(y, x)\}$, onde (1)

$d(x)^{i}$ é o tamanho do caminho da origem até o nó x.
i é o número de iterações.
d(y) é o tamanho do caminho da origem até o nó fechado (y).
d(y , x) é o tamanho do arco (y , x).

Atribui-se um valor d(x) para cada um dos nós do grafo, sendo:
d(origem) = 0
d(os outros nós) = ∞
Considerar "y" o último nó rotulado (fechado).
No início do algoritmo o nó de origem é o único rotulado, ou seja y = origem.

2º passo: para cada nó x não fechado, redefine-se d(x) conforme expressão 1. O nó aberto que possuir o menor valor d(x) é fechado e faz-se y = x.
3º passo: se o nó de destino foi fechado então se deve parar a execução do algoritmo, senão, volte ao passo 2.
Observação: para determinar a seqüência de nós que forma o caminho com distância mínima, deve-se, retroceder a partir do nó de saída, procurar os nós com etiquetas definitivas cuja diferença é igual à distância associada ao arco que os une.

Exemplo: Utilizando o grafo a seguir, identifique o seu caminho mínimo utilizando o algoritmo de Dijkstra:



1. $d(O) = 0$ e $d(1), d(2), d(3), d(4), d(D) = \infty$
2. $d(O) \to y = O$

i = 1, ou seja, 1ª iteração.

$d(1)1 = \min\{d(1)0, d(O) + d(O,1)\} = \min\{\infty, 0+4\} = 4$
$d(2)1 = \min\{d(2)0, d(O) + d(O,2)\} = \min\{\infty, 0+3\} = 3$
$d(3)1 = \min\{\infty, 0+3\} = 3$
$d(4)1 = \min\{\infty, 0+\infty\} = \infty$
$d(D)1 = \min\{\infty, 0+\infty\} = \underline{\infty}$

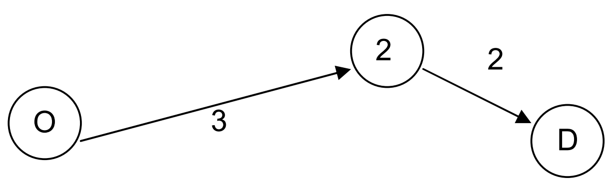
3. Identificar o mínimo entre as distâncias e definir y.
   Escolhe-se entre d(2) e d(3), pois esses apresentam atributos iguais a 3. Optou-se por y = 2 (nó fechado). Se y = D o problema está terminado, senão continuar do passo 2.

2. i = 2, ou seja, 2ª iteração.
d(1)2 = min{d(1)1, d(2) + d(2,1)} = min { 4, 3+∞} = 4
d(3)2 = min{d(3)1, d(2) + d(2,3)} = min { 3, 3+∞} = 3
d(4)2 = min{ ∞, 3+∞} = ∞
d(D)2 = min{ ∞, 3+2} = **<u>5</u>**
3. Mínimo entre as distâncias 4,3, ∞ e 5 é 3, ou seja, y = 3. O nó y é diferente de D, então continuar do passo 2.

2. i = 3, ou seja, 3ª iteração.
d(1)3 = min{d(1)2, d(3) + d(3,1)} = min { 4, 3+∞} = 4
d(4)3 = min{d(4)2, d(3) + d(3,4)} = min { ∞, 3+3} = 6
d(D)3 = min{ 5, 3+∞} = **<u>5</u>**
3. Mínimo entre as distâncias 4,6 e 5 é 4, ou seja, y = 1. O nó y é diferente de D, então continuar do passo 2.

2. i = 4, ou seja, 4ª iteração.
d(4)4 = min{d(4)3, d(1) + d(1,4)} = min {6, 4+2} = 6
d(D)4 = min{ 5, 4+∞} = **<u>5</u>**
3. Mínimo entre as distâncias 6 e 5 é 5, ou seja, y = D. O nó y agora é igual a D, então deve-se parar o processo de avaliação.

Pergunta-se:
Em qual iteração foi encontrado o primeiro valor de D (d(D) = 5)? Na 2ª iteração.
Qual era o valor de y nessa iteração? Na 2ª iteração, y é igual a 2.
Identificou-se o nó anterior ao destino: nó 2.
Em qual iteração foi encontrado o primeiro valor de 2 (d(2) = 3)? Na 1ª iteração.
Qual era o valor de y nessa iteração? Na 1ª iteração, y é igual a O.



<u>3. FLUXO MÁXIMO</u>

Neste tópico deve-se examinar um grafo orientado como uma Rede de Fluxo usando-a para analisar o fluxo de materiais a partir de uma origem, onde o material é produzido ou retirado, até um destino, onde o material é consumido ou depositado. A origem produz o material a uma taxa fixa e o depósito consome o material na mesma taxa. O "fluxo" do material em qualquer ponto no sistema é intuitivamente a taxa na qual o material se move.

Cada aresta orientada pode ser imaginada como um canal, com uma capacidade estabelecida, com uma taxa máxima na qual o material pode fluir pelo canal. Os vértices são junções de canais, onde o material flui sem acumulação. Isto é, com exceção da origem e do destino, a taxa de entrada e de saída de material no vértice deve ser a mesma. Chamamos essa propriedade de "conservação do fluxo".

Deseja-se então calcular a maior taxa na qual o material pode ser enviado da origem até o destino, sem violar as capacidades máximas das arestas e mantendo a propriedade de conservação de fluxo.

Uma Rede de Fluxo G(V,A) é um grafo orientado em que cada aresta $(u,v) \in A$ tem uma capacidade C(u,v) >= 0 (não negativa). Se uma dada aresta não está em A, então se supõe que a sua capacidade é zero (tais arestas não são desenhadas nos grafos). Numa rede de fluxo tem-se dois vértices especiais, uma origem "O" e um destino "D", e para todo vértice do grafo existe um caminho a partir de O passando por V que chega em D.

3.1. Método Ford-Fulkerson

O método de Ford-Fulkerson objetiva encontrar um fluxo máximo para uma rede de fluxos. É chamado de método por englobar diversas implementações com diferentes tempos de execução. O método é iterativo, começando com f(u,v) = 0.

Este método é composto pelos seguintes passos:

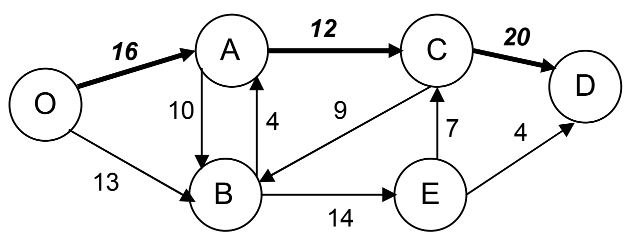
1º passo: iniciar o fluxo f total com 0 e verificar a existência de caminhos de fluxo > 0.

2º passo: Escolher um caminho da origem até o destino com fluxo >0; identificar o fluxo mínimo entre os fluxos presentes nos arcos (u,v) pertencentes ao caminho escolhido e para todas as arestas pertencentes ao caminho escolhido fazer:

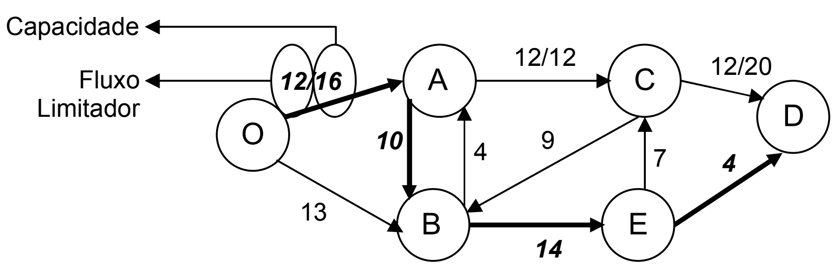
- ✓ f(u,v) = f(u,v) – f (decrementa o fluxo disponível)
- ✓ f(v,u) = f(v,u) + f (incrementa o fluxo utilizado)

3º passo: Faz-se $f_{total} = f_{total} + f$. O processo deve ser repetido até que todos os caminhos sejam analisados e enquanto existirem fluxos disponíveis.

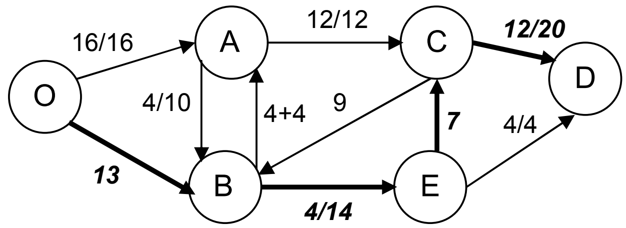
Exemplo: Baseando-se no grafo a seguir, identifique o fluxo máximo que pode fluir entre a origem (O) e o destino (D), utilizando o método de Ford-Fulkerson.



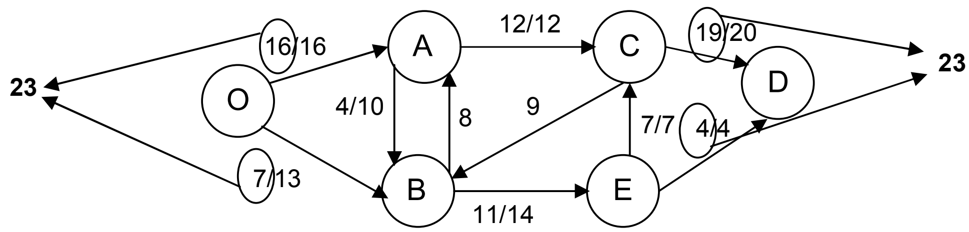
1º caminho escolhido: O>16>A>**12**>C>20>D, sendo f=12 e f_{total}=12



2º caminho escolhido: O>**4**>A>10>B>14>E>**4**>D, sendo f=4 e f_{total}=16



3º caminho escolhido: O>13>B>10>E>**7**>C>8>D, sendo f=7 e f_{total}=23



BIBLIOGRAFIA

Campos, Vânia B.G., **Otimização do Transporte**, Instituto Militar de Engenharia, Rio de Janeiro, 1998.