O Problema do Caminho de Custo Mínimo

Preâmbulo

O problema do Caminho de Custo Mínimo (PCM) é um problema muito estudado em teoria de grafos. As aplicações deste problema são inúmeras: trajetos rodoviários, rotas aéreas, encaminhamento de mensagens em telecomunicações, etc. Neste documento apresenta-se uma formulação em programação linear inteira para o PCM.

1. Noção de Caminho

Considere-se um grafo orientado, $G = (V, A)$, onde V denota o conjunto dos nós e A o conjunto dos arcos. Sejam s e t dois nós de G. Um caminho de s para t é uma sequência de arcos entre s e t em que para cada par de arcos consecutivos o nodo final do primeiro arco coincide com o vértice inicial do segundo e onde cada nó é visitado, no máximo uma vez.

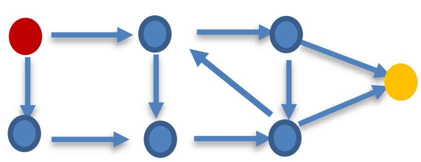	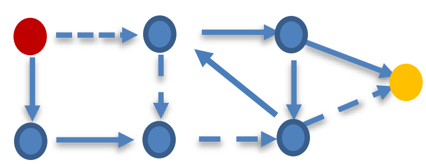
Figura 1A - Exemplo de um grafo dirigido, G1. O nodo ***s*** está assinalado a encarnado e o nodo ***t*** a amarelo.	Figura 1B – A sequência de arcos a tracejado é um exemplo de um caminho de ***s*** para ***t***.

Se a cada arco $(i, j) \in A$ estiver associado um custo, c_{ij}, calculamos o custo dum caminho somando os custos dos arcos nele incluídos. Nestas condições, o caminho de custo mínimo entre ***s*** e ***t*** será o caminho de ***s*** para ***t*** para o qual a soma dos custos dos arcos é mínima. No exemplo da Figura 1, se assumirmos que que $c_{ij} = 1, \forall (i, j) \in A$, o caminho a tracejado tem custo 4. Verifica-se, por observação direta que o caminho a tracejado não é o caminho de custo mínimo, pois é possível definir em G1 pelo menos um caminho com custo 3. Para saber se esse será ou não o caminho de custo mínimo seria comprovar que não existe nenhum outro menos do que ele. Realizar essa tarefa enumerando todos os caminhos é um processo nada eficiente. Então, interessa-nos identificar um procedimento que permita determinar o caminho de custo mínimo de forma sistemática em qualquer rede orientada, G.

2. Formulação em Programação Linear Inteira do Problema de Caminho de Custo Mínimo

M. T Godinho, IPBeja, 2024

Considerem-se variáveis binárias x_{ij} , associadas aos arcos de G, e que para cada $(i,j) \in A$ tomam o valor 1 se o arco (i,j) estiver no caminho e 0 caso contrário. Então, o Problema do Caminho de Custo Mínimo pode ser formulado em Programação Linear Inteira como se segue:

$$\min Z = \sum_{(i,j)\in A} c_{ij}x_{ij} \qquad (1)$$

s.a.:

$$\sum_{j:(s,j)\in A} x_{sj} = 1 \qquad (2)$$

$$\sum_{i:(i,t)\in A} x_{it} = 1 \qquad (3)$$

$$\sum_{i\neq t:(i,j)\in A} x_{ij} - \sum_{i\neq s:(j,i)\in A} x_{ji} = 0 \qquad j \in V\backslash\{s,t\} \qquad (4)$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in A \qquad (5)$$

3. Algoritmos para o cálculo do caminho de custo mínimo numa rede.

Para determinar o caminho mais curto entre um vértice s e um vértice t pode ser aplicado o algoritmo de Dijkstra, caso a rede não contenha arcos com custos negativos, ou o algoritmo de Floyd, em qualquer caso. Na maior parte das situações os custos associados aos arcos são não negativos, pelo que o Algoritmo de Dijkstra é o mais usado no estudo de problemas desta natureza. Existem várias representações em pseudocódigo do Algoritmo de Dijkstra, todas elas equivalentes. No nosso trabalho usaremos a versão disponibilizada em [2] (v. Figura 2). Nesta versão, os autores introduzem uma ligeira modificação ao algoritmo original, tornando-o mais eficiente. O algoritmo está descrito para o caso em que os custos são identificados com o comprimento dos arcos, mas pode ser adaptado a qualquer outro tipo de custos.

Modified Dijkstra Algorithm for Shortest Path from A to Z

The algorithm given below is a slight variant of the original Dijkstra algorithm [3,4]. It is different (in Step 3 below) in that it scans all the neighbors of the vertex selected (or "permanently" labeled) in Step 2.

Let d(i) denote the distance of vertex i from starting vertex A. Let P(i) denote its predecessor.

1. Start with
   $d(A)=0$, $d(i)=l(A,i)$, if $i \varepsilon \Gamma_A$,
   $=\infty$, otherwise.
   ($\Gamma_i \equiv$ set of first neighbor vertices of vertex i,
   $l(i,j) =$ length of arc from vertex i to vertex j)
   $P(i)=A\ \forall\ i \varepsilon \Gamma_A$.
   Set $\bar{S}=\Gamma_A$

2. Find $j \varepsilon \bar{S}$ such that $d(j)=\min d(i),\ i \varepsilon \bar{S}$.
   Set $\bar{S}=\bar{S}-\{j\}$
   If j = Z (the terminal vertex), END;
   otherwise, go to 3.

3. $\forall\ i \varepsilon \Gamma_j$, if $d(j)+l(j,i) < d(i)$, set
   $d(i)=d(j)+l(j,i)$, $P(i)=j$ and $\bar{S}=\bar{S}\cup\{i\}$;
   go to 2.

Figura 2 – O Algoritmo de Dijkstra na versão apresentada em [2].

No pseudocódigo acima, para cada vértice i, $d(i)$ denota a distância da origem (vértice A) a i e $P(i)$ denota o seu predecessor no caminho em análise. Por exemplo, no caminho representado na Figura 3:

1 –> 7 –> 34 -> 14 -> 8

Figura 3 – Exemplo de um caminho entre uma origem e um destino. Assumindo que os custos são todos unitários, teríamos, por exemplo, d(14)=3 e P(14)=34.

Referências

[1] Ahuja, R. K., Magnanti, T. L., Orlin, J. B. (1993). *Network Flows: Theory, Algorithms, and Applications*. Prentice hall.

[2] R. Bhandari, "Optimal physical diversity algorithms and survivable networks (1997)" *Proceedings Second IEEE Symposium on Computer and Communications*, Alexandria, Egypt, pp. 433-441, doi: 10.1109/ISCC.1997.616037.[1] Optimal physical diversity algorithms and survivable networks | IEEE Conference Publication | IEEE Xplore