CAPÍTULO II

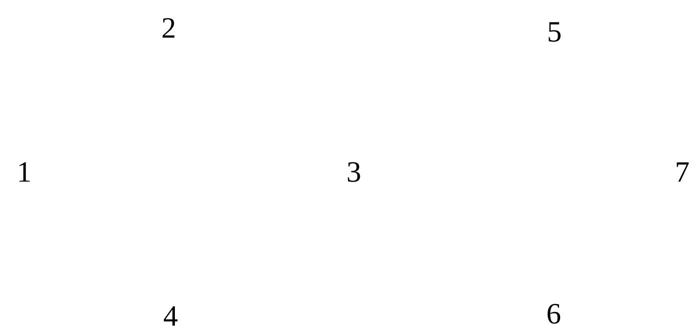
ÁRVORE GERADORA MÍNIMA

Deseja-se construir uma rede de comunicação entre várias cidades à custo mínimo. Sabe-se que o custo de qualquer ligação é dado por c_{ij}..

Representando esta rede por um grafo conexo $G = (N, A)$, onde N é o conjunto de cidades e A as ligações entre elas, e associando a cada arco $a_j \in A$ o custo $c(a_j)$, então o problema consiste em determinar o menor grafo parcial conexo de custo mínimo, ou a árvore geradora mínima de G.

O problema acima pode ser resolvido escolhendo os arcos em ordem crescente dos $c(a_j)$, rejeitando um arco somente se o mesmo forma um ciclo com os já selecionados.

EXEMPLO: Seja o grafo $G = (N, A)$ da figura abaixo, onde $N = \{1, 2, 3, 4, 5, 6, 7\}$.



A árvore geradora mínima é representada por traços cheios na figura. Um conjunto composto de seis arcos uma vez que árvore geradora deve conter $|N|$ - 1 arcos

TEOREMA: Uma condição necessária e suficiente para que $G' = (N, A')$ seja uma árvore *geradora mínima*, é que para todo arco $a_u \in (A - A')$, o ciclo A'', $A'' \subset A' \cup \{a_u\}$, verifique:

$$c(a_u) \geq c(a_v),\ \forall\ a_v \in A''\ (a_v \neq a_u)$$

A demonstração pode ser vista em *Gondran e Minoux* (1983).

O algoritmo utilizado acima caracteriza-se por escolher, a cada passo, um arco de peso mínimo que anexado aos elementos até então selecionados forneça o menor acréscimo possível até que seja encontrado um conjunto (*árvore geradora*) solução do problema. Volta-se para o acréscimo relativo e não para o valor absoluto do objetivo. Esses algoritmos são denominados de **GULOSOS**.

1 . ALGORITMOS

Seja um conjunto finito $A = \{a_1, a_2, ..., a_m\}$ e uma função de conjuntos f, definida sobre os subconjuntos de A. Os algoritmos *gulosos* procuram um subconjunto $S^* \subseteq A$ tal que:

$$f(S^*) = \min_{S \subseteq A} f(S) \quad \text{onde} \quad f(S) = \sum_{a_j \in S} c(a)$$

Algoritmo Guloso Geral

k = 0;

$S^k = \varnothing$;

ordene os elementos de *A* de modo que: $c(a_1) \leq c(a_2) \ldots \leq c(a_m)$;

repita

$$S^{k+1} \longleftarrow \begin{cases} S^k \cup \{a_{k+1}\}, & \text{se "CONDIÇÃO" é satisfeita;} \\ S^k & \text{caso contrário;} \end{cases}$$

$k \longleftarrow k+1$;

até k = m;

fimalgoritmo;

Para o problema de árvore geradora mínima, a "CONDIÇÃO" do algoritmo é satisfeita, se o arco selecionado não implica na formação de um ciclo.

O primeiro algoritmo guloso surgiu da demonstração do teorema (*Boruvka*, 1926):

"Se a um grafo conexo finito associamos um número real positivo, ou um peso a cada arco, e se estes números são todos distintos, então existe uma única árvore geradora cuja soma dos pesos em seus arcos é mínima dentre todas as árvores geradoras possíveis".

Este teorema é demonstrado utilizando um método de construção de árvore geradora mínima bastante complexo. Com a teoria dos grafos, *kruskal* (1956) demonstrou este teorema apresentando o método construtivo enunciado acima, e formalizado abaixo em três versões para um grafo *G* = (*N*, *A*).

Algoritmo de *Kruskal* (1)

ordene os arcos em *A* em ordem crescente de custos:

$$c(a_1) \leq c(a_2) \leq \ldots \leq c(a_m);$$

$S^* \leftarrow \{a_1\}$;

$k \leftarrow 1$;

repita

$k \leftarrow k+1$;

se a_k "não forma ciclo com os arcos de S*" então faça $S^* \leftarrow S^* \cup \{a_k\}$;

até $k = m$;

fimalgoritmo.

Algoritmo de Kruskal (2) (Arcos não ordenados)

$S^* \leftarrow \{a_1\}$;

$k \leftarrow 1$;

repita

$k \leftarrow k+1$;

$S^* \leftarrow S^* \cup \{a_k\}$;

se S* "contém um ciclo *A'* "

então seleciona arco $a_v \in A'$ de custo máximo;

faça $S^* \leftarrow S^* - \{a_v\}$;

até $k = m$;

fimalgoritmo.

Algoritmo de Kruskal (3)

ordene os arcos em *a* em ordem decrescente de custos:

$$c(a_1) \geq c(a_2) \geq \ldots \geq c(a_m);$$

$S^* \leftarrow A;$

$k \leftarrow 0$;

repita

$k \leftarrow k + 1;$

se $G' = (N, S^* - \{a_k\})$ "é conexo" então faça $S^* = S^* - \{a_k\}$;

até $k = m;$

fimalgoritmo.

Em todos os algoritmos o grafo parcial $G' = (N, S^*)$ é a árvore geradora mínima. Nos dois primeiros casos, a árvore final é obtida pela junção de pequenas árvores de uma floresta que inicialmente contém n árvores triviais, com único vértice. Quando todas as árvores da floresta tiverem sido fundidas em um única, contendo $(n - 1)$ arcos, resulta o grafo parcial G'.

A otimalidade dos algoritmos acima pode ser mostrada pelo processo construtivo, conforme Kruskal (1956), Szwarcfiter (1984). No entanto, uma outra alternativa é demonstrá-la tendo como base a teoria de matróides.

2 .ÁRVORE GERADORA MÍNIMA E A TEORIA DE MATRÓIDES

A teoria de matróides surgiu da tentativa de unificar, de certa maneira, a álgebra e a teoria dos grafos. Hassley Whitney após vários anos de estudos sobre a teoria dos grafos, notou similaridades entre as idéias de independência linear e posto em teoria dos grafos, e também entre a independência linear e a dimensão no estudo de espaços vetoriais. Em seu paper básico, Whitney (1935) usa o conceito de matróides para formalizar essas similaridades.

Sejam $A_1, A_2, ..., A_m$ colunas de uma matriz A. Qualquer subconjunto destas colunas é linearmente independente ou dependente. Nota-se que os subconjuntos independentes satisfazem as seguintes propriedades:

(a) Qualquer subconjunto de um conjunto independente é também independente;

(b) Se S_p e S_{p+1} são conjuntos independentes com p e $p + 1$ elementos respectivamente, então S_p com alguma coluna de S_{p+1} forma um conjunto independente com $p + 1$ elementos.

Whitney verificou que outros sistemas, não representados por matrizes, satisfaziam as mesmas propriedades. A esses sistemas ele denominou um matróide. Trata-se portanto de uma generalização das propriedades de matriz. Assim, um matróide é essencialmente um conjunto sobre o qual é definido uma “estrutura de independência”, uma generalização dos conceitos de independência e dependência linear.

MATRÓIDE $M = (E, F)$ é uma estrutura em que E é um conjunto finito de elementos e F é uma família de partes de E que verificam:

(a) $\emptyset \in F$

(b) $S \in F$ e $R \subseteq S$ então $R \in F$

(c) Se R e S pertencem a F com $|S| = |R| + 1$ então existe $s \in (S - R)$ tal que $(R \cup \{s\}) \in F$.

Por analogia à Álgebra Linear, os elementos de F são denominados subconjuntos independentes de E, satisfazendo a “estrutura de independência” definida sobre E. Uma independência geral que pode ser bastante diferente da clássica independência linear.

Com isso, podem ser definidos vários tipos de matróides: vetorial, matricial, gráfico, uniforme, transversal, etc.

Seja um grafo $G = (N, A)$. Seja $E = A$, o conjunto de arcos de G. Seja F o conjunto de subconjuntos de E tal que para todo $S \subseteq E$, $S \in F$ se só se S não contém um ciclo de G. O matróide $M = (E, F)$, assim definido, é denominado *GRÁFICO*. Ainda mais, uma árvore geradora de G é um subconjunto independente do matróide M.

Os matróides estão intimamente ligados aos algoritmos gulosos. Isto porque, para o caso específico de matróides, estes algoritmos são exatos e fornecem soluções ótimas. Neste caso, a “condição” expressa no algoritmo guloso geral, apresentado anteriormente, é dada pela “estrutura de independência” definida sobre o matróide. Basta verificar se $(S^k \cup \{a_{k+1}\}) \in F$.

TEOREMA: Seja o conjunto finito E e F um conjunto de subconjuntos de E. Se $M = (E, F)$ é um matróide, então para toda função de custos não negativos c: $E \leftarrow \Re$ o guloso encontra o máximo subconjunto independente de custo mínimo do matróide, (S^*).

A demonstração pode ser vista em Mateus (1980).

Com os resultados anteriores, o problema de árvore geradora mínima é resolvido definindo o matróide gráfico associado ao grafo $G = (N, A)$ e aplicando o algoritmo guloso de Kruskal. A otimalidade é garantida pelo teorema anterior.

Um outro algoritmo guloso para determinação da árvore geradora mínima foi proposto por Prim (1957). A verificação da formação de ciclos está embutida na operacionalização do algoritmo. Separa-se os nós do grafo em dois conjuntos: P, conjunto de nós pesquisados e $\overline{P}$ conjunto dos nós não pesquisados. Define-se um ARCO MINIMAL de j $\in$ $\overline{P}$ ao arco de custo mínimo ligando j a algum nó i $\in$ P. O algoritmo inicia o processo por um nó arbitrário. Seja i este nó. Então, P = {i} e $\overline{P}$ = N - {i}. Determina-se os arcos minimais para todo $j \in \overline{P}$. Como P contém um único elemento, estes arcos serão então todos os arcos contendo i como uma das extremidades. Aos nós $j \in \overline{P}$ não estão ligados diretamente a i, associa-se um arco artificial com peso infinito. A seguir, determina-se como novo elemento de P o nó $j \in \overline{P}$ correspondente ao menor arco minimal. Seja k este nó, $k \in \overline{P}$. Faz-se $P = P \cup \{k\}$ e $\overline{P} = \overline{P} - \{k\}$. Atualiza-se os arcos minimais de $j \in \overline{P}$. Seqüencialmente, escolhe-se um novo elemento para P sucessivamente até $\overline{P} = \emptyset$. Como no início do processo, $\left|\overline{P}\right| = n-1$, $(n-1)$ arcos serão selecionados formando uma árvore geradora.

Algoritmo de Prim

Seja o nó inicial $i = 1$. Seja $L(j)$ o vetor contendo para cada $j \in \overline{P}$ a extremidade em P do arco minimal de j, ou seja, $L(j) = i$, onde $i \in P$ e (i, j) é o arco minimal. Seja

$n = |N|$.

para $j = 2, ..., n$ faça

$L(j) \leftarrow 1$;

$$c_j \leftarrow \begin{cases} c_{ij} & \text{se existe o arco } (1, j) \in A \\ \infty, & \text{caso contrário;} \end{cases}$$

fimpara;

$P \leftarrow \{1\}$, $\overline{P} \leftarrow \{2, ..., n\}$, $S^* = \emptyset$;

repita

selecione $k \in \overline{P}$ tal que $c_k = \min_{j \in \overline{P}} \{c_j\}$;

faça $P \leftarrow P \cup \{k\}$;

$\overline{P} \leftarrow \overline{P} - \{k\}$;

$S^* \leftarrow S^* \cup \{L(k), k\}$;

se $\overline{P} \neq \emptyset$ então para todo $j \in \overline{P}$ faça

se $c_{kj} < c_j$ então faça

$c_j \leftarrow c_{kj}$;

$L(j) \leftarrow k$;

fimse;

fimpara;

fimse;

até $\overline{P} = \emptyset$;

fimalgoritmo.

O conjunto de arco $S^* \subseteq A$ é uma árvore geradora mínima. É fácil verificar que todos os arcos de custo mínimo selecionados pelo algoritmo de Kruskal (1), em ordem crescente, são selecionados pelo algoritmo de *PRIM* em um ordem não necessariamente crescente, obtendo a mesma solução. Trata-se portanto, de um algoritmo guloso exato para o problema em estudo.

EXEMPLO: Seja o grafo da figura apresentada no exemplo anterior. Aplicando o algoritmo de *PRIM* temos:

$L(2) = L(3) = ... = L(7) = 1$

$c = (c_2, c_3, ..., c_7) = (2, 4, 8, \infty, \infty, \infty)$

$P = \{1\}$, $\overline{P} = \{2, 3, ..., 7\}$, $S^* = \emptyset$

$c_k = \min \{2, 3, 4, 8, \infty\} = 2 \rightarrow k = 2$

$P = \{1, 2\}$, $\overline{P} = \{3, 4, ..., 7\}$, $S^* = \{(1, 2)\}$

$c_{23} = 10 > 4 = c_3$

$c_{24} = \infty > 8 = c_4$

$c_{25} = 12 < \infty = c_5 \rightarrow c_5 = 12, L(5) = 2$

$c_{26} = c_{27} = c_6 = c_7 = \infty$

$c_k = \min \{4, 8, 12, \infty, \infty\} = 4 \rightarrow k = 3$

$P = \{1, 2, 3\}$, $\overline{P} = \{4, 5, 6, 7\}$, $S^* = \{(1, 2), (1, 3)\}$

O processo prossegue selecionando os seguintes arcos:

(1, 2), (1, 3), (4, 6), (3, 7), (7, 5).