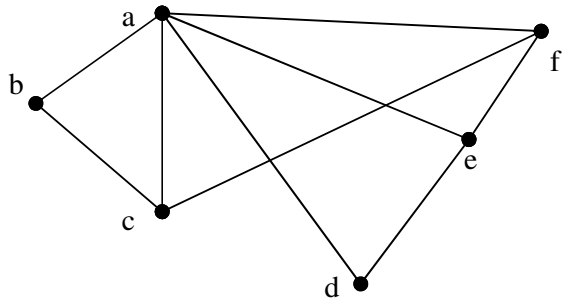
11.4 Árvores Geradoras

Em muitas aplicações, estamos interessados em subgrafos especiais de um determinado grafo.

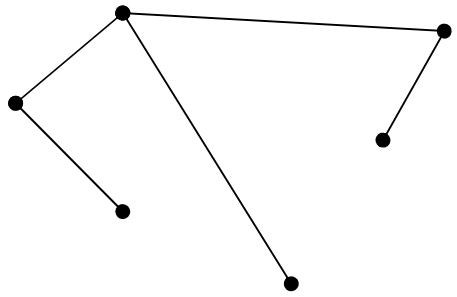
Definição – Árvore Geradora - uma árvore T é chamada de árvore geradora de um grafo G se T é um subgrafo de G que possui todos os vértices de G.

Exemplo 1:

G:



T_1:



Obs: Uma submatriz (n-1) x (n-1) da matriz de incidência é não singular se e somente se as arestas associadas às (n-1) colunas desta submatriz constituem uma árvore geradora de G.
Posto de um grafo – o posto de uma grafo com *n* vértices é igual a *n-1*.

Como obter uma árvore geradora de G?
Procedimento 1- Se G não possui circuitos, G é sua própria árvore geradora. Se G possui circuitos, retire uma aresta do circuito. O sugrafo resultante é conexo. Se existirem mais circuitos, repita a operação até retirar uma aresta do último circuito do grafo. O subgrafo resultante é conexo, sem circuitos e possui todos os vétices de G. Portanto é uma árvore geradora de G.

Teorema – Todo grafo conexo contém pelo menos uma árvore geradora.

Definição – Seja G(V,A) um grafo conexo e T(V,E) uma árvore geradora de G. Uma aresta de G que não pertence à árvore geradora T é chamada de **elo** de G em relação a T. As arestas que compõem uma árvore geradora são chamadas de **ramo.**

Exemplo 2: Os elos de G relativos a T_1 na exemplo 1 são: {(c,f). (a,c),(d,e),(a,e)}.

Obs: Observe que uma aresta que pertence a T_1 pode ser um elo de G em relação a uma outra árvore geradora de G. No entanto, o numero de elos de um grafo é fixo!

Quantos são?

Teorema – Um grafo conexo com *n* vértices e *m* arestas possui $(m - n + 1)$ elos.

Se adicionarmos um elo de G a árvore T_1, um único circuito será formado. Este circuito é chamado de circuito fundamental de G. Quantos circuitos fundamentais um grafo possui?

Dado um grafo G, como obter todas as árvores geradoras de G?

Procedimento 2 –
1 - Utilize o procedimento 1 para obter uma árvore geradora inicial.
2 - Determine os elos de G relativos a esta árvore. Acrescentando um elo de G a T_1 um circuito é formado.
3 - Retire as arestas do circuito fundamental formado uma a uma. Desta foram são geradas as arvores geradoras associadas às arestas deste circuito, (*k-1*) árvores geradoras, onde *k* é o número de arestas no circuito fundamental. Esta operação é chamada de transformação elementar (troca cíclica, *cyclic exchange*).
4 - Repita a transformação elementar considerando outros elos do grafo.

A análise do Procedimento 2 esboçado acima, permite a formulação de uma série de perguntas.
1) Partindo de qualquer árvore e fazendo um certo número de transformações elementares é possível obter uma determinada árvore geradora?
2) Usando transformações elementares é possível obter todas as árvores geradoras? Quantas transformações elementares serão necessárias?
3) A eficiência do algoritmo depende da árvore geradora inicial?

Para responder algumas dessas perguntas precisamos definir alguns novos conceitos.

Distância entre duas árvores geradoras de G, T_i e T_j, é igual ao número de arestas que estão presentes em T_i e que não pertencem a T_j. Denotamos por $d(T_i,T_j)$. Podemos definir a distâcia entre duas árvores geradoras de G como sendo o número de transformações elementares necessárias para obter T_j a partir de T_i. Isto é:

$$d(T_i,T_j) = \frac{1}{2}\left|A_{T_i \oplus T_j}\right|.$$

Árvore central – Para uma árvore geradora T_0 de G, seja $\max_i d(T_0,T_i)$ a distância máxima de T_0 a qualquer outra árvore geradora T_i de G. Então T_0 é chamada de árvore central de G se:

$$\max_i d(T_0,T_i) \le \max_j d(T\ ,T_j), \forall T \text{ árvore geradora de G.}$$

Grafo árvore – O grafo árvore de G, S(G), é definido como o grafo em que cada vértice representa uma árvore geradora de G e existe uma aresta entre dois pares de vértices a distância entre as árvores geradoras associados for igual a 1.

Estes conceitos são usados em [2] para encontrar todas as árvores geradoras de G. A Figura 1 mostra um grafo G_1 e o grafo árvore, $S(G_1)$ associado. Um outro algoritmo para listar todas as árvores geradoras de G é proposto em [3].

Teorema – É possível gerar todas as árvores geradoras de G começando de uma árvore geradora qualquer e executando sucessivas transformações elementares.

Exemplo

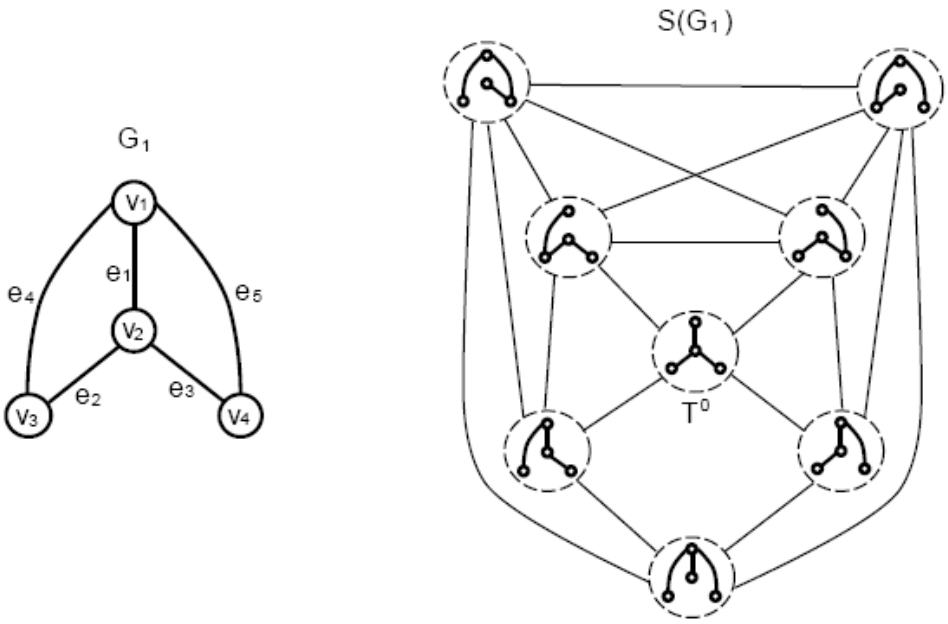


Figura 1 - Grafo G e Grafo Árvore associado S(G)[2]

O Procedimento 1 constrói uma árvore geradora de G através da exclusão de arestas que fazem parte de um circuito em G. O algoritmo 1 a seguir constrói uma árvore geradora de G incluindo arestas evitando a formação de circuitos.

Algoritmo 1 - Determinar uma árvore geradora

Considere um grafo simples com *n* vértices e *m* arestas.

Idéia – Inicie a árvore T com uma aresta qualquer de G. A cada iteração, inclua uma nova aresta em T de maneira que nenhum circuito é formado.

1) O que acontece se o grafo não for conexo? Iremos obter várias árvores geradoras, isto é uma floresta geradora.

2) Como garantir que ao inserir uma aresta nenhum circuito é formado? Verificar se as extremidades da aresta já foram incluídas. Assim ao tentarmos acrescentar a aresta (v_k, w_k) à árvore, as seguintes situações podem ocorrer:
i) Nem o vértice v_k, nem o vértice w_k pertencem a alguma árvore T_i já construída. Neste caso crie uma nova árvore a partir destes vértices e desta aresta. Considere que existe mais de uma componente no grafo, faça $cp = cp+1$. Associe o rótulo *cp* aos vértices v_k e w_k.

ii) O vértive v_k pertence à árvore T_i e o vértice w_k pertence à árvore T_j, $i \neq j$. Neste caso, a aresta (v_k, w_k) é usada para unir as duas árvores. Faça os vértices de T_j receberem o mesmo rótulo *c* dos vertices de T_i. Faça $cp=cp-1$.

iii) Os dois vértices pertencem a árvore T_i.. Neste caso a aresta é descartada pois sua inclusão criaria um circuito.

iv) Apenas um dos dois vértices v_k (ou w_k) pertence a alguma árvore T_i já construída. Neste caso acrescente a aresta e o vértice w_k (ou v_k) à árvore. O vértice w_k (ou v_k) recebe o mesmo rótulo *c* que os vértices já pertecentes a T_i.

Como fazer para implementar as idéias acima?
A eficiência do algoritmo depende da rapidez com que verificamos se as extremidades da aresta que estamos considerando pertence ou não a alguma árvore já criada.

Para facilitar esta busca, criamos um vetor *n*-dimensional VERTEX que armazena esta informação. Quando uma aresta (i,j) é inserida em alguma árvore com rótulo *c*, as posições i e j do vetor recebem o valor *c*. Assim para verificar se a aresta (v_k , w_k) já foi incluído em alguma árvore, verificamos se correspondentes posições de VERTEX são diferentes de zero. Se para algum vértice q, VERTEX(q) = 0 o vértice q não esta incluído em nenhuma árvore. Ao final do algoritmo o vetor VERTEX identifica os vértices em cada componentes do grafo.

Isto é suficiente?
Precisamos ainda identificar as arestas que compõe cada árvore do grafo. Para isto criamos o vetor m-dimensional ARESTA. Assim se a k-ésima aresta foi incluída na árvore *c*, faça ARESTA (k) = *c*, caso contrário ARESTA (k)=0. Ao final do algoritmo, as posições do vetor com ARESTA(i)=0 identificam os elos de G.

Exemplo
Considere o grafo G com 9 vértices e 12 arestas. O grafo será representado através de dois vetores m-dimensionais F e H., de tal forma que as extremidades da aresta K é armazenada nas posições fk e hk dos vetores F e H respectivamente. G é dado por:

F =

A	E	I	I	B	B	C	B	F	C	F	A

H =

B	H	B	C	C	E	G	F	G	E	D	C

11. 5 - Árvore Geradora Mínima

Considere um rede e o problema de encontrar a árvore geradora mínima associada.

Valor de árvore - é a soma dos pesos associados ás arestas contidas na árvore.

Algoritmo 2 (Kruskal) - Grafos Conexos

Passo 1- ordene as arestas do grafo em ordem não-decrescente de peso.

Passo 2 - Aplique o algoritmo 1 para encontrar a arvore geradora, considerando que as arestas serão selecionadas de acordo com a ordem estabelecida no passo 1.

Lema – A árvore geradora T obtida pelo Algoritmo 2 é uma árvore geradora mínima de G.

Prova - Sejam e_1, e_2, ..., e_{n-1} as arestas de T na ordem em que foram consideradas no Algoritmo 2. Isto é $p(e_1) <= p(e_2) <= ... <= p(e_{n-1})$.

Suponha que T não é uma árvore geradora mínima de G. Seja T_{min} a árvore geradora que contém as arestas e_1, e_2, ..., e_j , tal que j seja o maior índice possível.

Considere que a aresta e_{j+1} é adicionada a T_{min}. Um circuito é então criado. Este circuito contém uma aresta x que não pertence a T (Se todas as arestas do circuito estivessem em T, T não seria uma árvore, pois também teria um circuito).

Pela ordem em que as arestas foram consideradas na construção de T, temos que e_{j+1} foi adicionada a T, mas x não foi incluída. Portanto $p(e_{j+1})<=p(x)$ (caso contrário x teria sido incluída em T sem a formação de um circuito).

Vamos então construir uma nova árvore:

$$T_{nova}=T_{min}-\{x\}+\{e_{j+1}\}.$$

Se $p(e_{j+1}) < p(x)$ então $p(T_{nova}) < p(T_{min})$ o que contraria a hipótese que T_{min} é mínima.

Se $p(e_{j+1}) = p(x)$ então $p(T_{nova}) = p(T_{min})$ e T_{nova} é mínima e contém as arestas e_1, e_2, ..., e_j, e_{j+1}, o que contradiz que j é o menor índice possível usado na construção de T_{min}.

Portanto, temos uma contradição quando dizemos que

$$p(e_{j+1})<=p(x)$$

e neste caso a suposição que T não é mínima é falsa. Assim mostramos que T é mínima.

Algoritmo 3 (PRIM) - Grafos Conexos

Passo 1 - Selecione um vértice v_k de G e inclua em T

Passo 2 - Repita este passo até que todos os vértices de G pertençam a T.

Selecione a aresta de menor peso (v_j, w_j) tal que v_j pertença a T e w_j não pertença a T.

Exercício [1] - Utilize os algoritmos de Kruskal e de Prim para identificar uma árvore geradora mínima em cada um dos grafos ilustrados na figura 1 e 2 . Qual é o melhor?

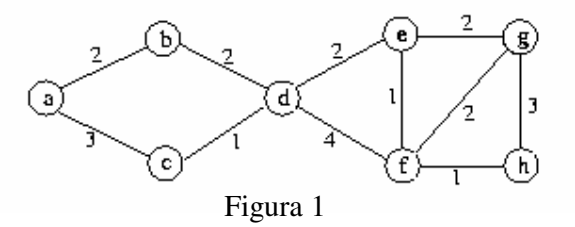


Figura 1

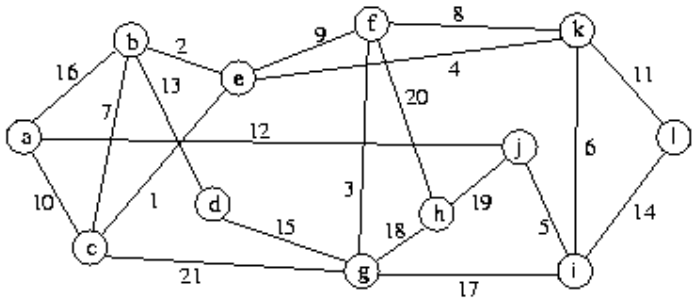


Figura 2

Referências$

[1] - Michel Gagnon - Notas de aula do curso: CI065 Algoritmos e teoria dos grafos UFPR, 2002.
[2] – Shioura, A., A. Tamura e T. Uno, An optimal algorithm for scanning all spanning trees of undirected graphs. *SIAM Journal on Computing*

[3] - A Simple Algorithm for Listing All the Trees of a Graph, Minty, G.,Circuits and Systems, *IEEE Transactions on Circuit Theory*, Volume 12, Issue 1, Mar 1965 Page(s): 120 – 120.