7.8.1 Algoritmo Genérico para Obter a Árvore Geradora Mínima

O algoritmo genérico apresenta uma estratégia gulosa que permite obter a árvore geradora mínima adicionando-se uma aresta de cada vez. O algoritmo gerencia um subconjunto S de arestas, mantendo o seguinte invariante para o anel:

> Antes de cada iteração, S é um subconjunto de uma árvore geradora mínima.

A cada passo determinamos uma aresta (u, v) que possa ser adicionada a S sem violar este invariante, já que $S \cup \{(u, v)\}$ é também um subconjunto de uma árvore geradora mínima. Tal aresta é chamada uma **aresta segura** para o subconjunto S, uma vez que ela pode ser adicionada a S e manter o invariante.

O Programa 7.17 apresenta o algoritmo genérico para obter uma árvore geradora mínima, de acordo com o algoritmo guloso genérico apresentado pelo Programa 2.11.

Programa 7.17 *Algoritmo genérico para obter a árvore geradora mínima*

```
   procedure GenericoAGM;
1  S := ∅;
2  while S não constitui uma árvore geradora mínima do
3     (u, v) := seleciona(A);
4     if aresta (u, v) é segura para S then S := S + {(u, v)}
5  return S;
```

O anel nas linhas 2-4 do Programa 7.17 mantém o invariante já que apenas arestas seguras são adicionadas a S. A parte importante é encontrar uma aresta segura na linha 3. Dentro do corpo do anel **while**, S tem de ser um subconjunto próprio da árvore geradora mínima T, e assim tem de existir uma aresta $(u, v) \in T$ tal que $(u, v) \notin S$ e (u, v) seja seguro para S.

Antes de apresentar uma regra para reconhecer arestas seguras, precisamos de algumas definições. Um **corte** $(V', V - V')$ de um grafo não direcionado $G = (V, A)$ é uma partição de V, conforme ilustra a Figura 7.17. Uma aresta $(u, v) \in A$ *cruza* o corte $(V', V - V')$ se um de seus vértices pertence a V' e o outro vértice pertence a $V - V'$. Ao lado de cada vértice é mostrada a cor branca ou preta (b ou p). Os vértices em V' são vértices pretos e os vértices em $V - V'$ são vértices brancos. As arestas cruzando o corte são aquelas que conectam vértices brancos a vértices pretos. Um corte *respeita* um conjunto S de arestas se não existirem arestas em S que cruzem o corte. Uma aresta que tenha custo mínimo sobre todas as arestas cruzando o corte é uma *aresta leve*. Note que pode haver mais de uma aresta leve cruzando um corte. De modo geral, uma aresta satisfazendo determinada propriedade é leve se tiver peso mínimo sobre qualquer aresta que satisfaça a propriedade.

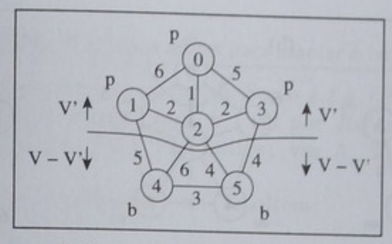

Figura 7.17 *Um corte $(V', V - V')$ do grafo $G = (V, A)$ da Figura 7.16(a).*

O teorema a seguir apresenta uma regra para reconhecer arestas seguras. A prova do teorema pode ser encontrada em Cormen, Leiserson, Rivest e Stein (2001, p. 563).

Teorema 7.8.1: Considere $G = (V, A)$ um grafo conectado, não direcionado e com pesos p sobre as arestas V. Considere S um subconjunto de V incluído em alguma árvore geradora mínima para G, considere $(V', V - V')$ um corte qualquer que respeita S, e considere (u, v) uma aresta leve cruzando $(V', V - V')$. Logo, a aresta (u, v) é uma aresta segura para S.

7.8.2 Algoritmo de Prim

O algoritmo de Prim para obter uma árvore geradora mínima pode ser derivado do algoritmo genérico apresentado no Programa 7.17. Ele utiliza uma regra específica para determinar uma aresta segura na linha 3 do Programa 7.17. Nesse caso, o subconjunto S forma uma única árvore, e a aresta segura adicionada a S é sempre uma aresta de peso mínimo conectando a árvore a um vértice que não esteja na árvore. De acordo com o Teorema 7.8.1, apenas arestas seguras para S são adicionadas, o que significa que, quando o algoritmo termina, as arestas em S formam uma árvore geradora mínima.

A Figura 7.18 ilustra a execução do algoritmo de Prim sobre o grafo da Figura 7.16(a). No início do processamento, todos os vértices são iniciados com peso igual a infinito. Arestas em negrito pertencem à árvore sendo construída. A cada passo do algoritmo, os vértices na árvore determinam o corte no grafo, e uma aresta leve cruzando o corte é adicionada à árvore S, conectando S a um vértice de $G_S = (V, S)$. A árvore começa pelo vértice 0. Na realidade, o algoritmo poderia começar por um vértice arbitrário Raiz qualquer. Ao escolher o vértice 0 para iniciar a árvore, o corte separa esse vértice dos vértices 1, 2 e 3, cujos pesos passam a ser 6, 1 e 5, respectivamente. Conforme ilustra Figura 7.18(b), a árvore cresce a partir do vértice 0 ao escolher a aresta de menor peso cruzando o corte, no caso a aresta $(0, 2)$. Essa etapa é repetida até que a árvore "gere" todos os vértices em V. No segundo passo do algoritmo, mostrado na Figura 7.18(c), existe a escolha de adicionar a aresta $(2, 1)$ ou a aresta $(2, 3)$ à árvore, uma vez que ambas são arestas leves cruzando o corte.