

Capítulo VI

Identificação de Restrições Violadas

VI.1 - Introdução:

Neste capítulo, será discutida nossa abordagem que trata da identificação de restrições violadas a partir da solução de um problema lagrangeano, que consiste, basicamente, na determinação de uma K -árvore mínima $c/2K$ arestas incidentes ao depósito. A identificação de restrições violadas neste caso, é significativamente mais simples se comparada à relaxação linear (onde temos soluções podendo assumir valores fracionários).

Não é conhecido até o momento, nenhum algoritmo exato polinomial para a separação de sub-rotas violadas por uma relaxação linear (associada a uma formulação do problema de roteamento). Como Harche e Rinaldi[91] mostraram que este problema de separação é NP-completo, sua resolução em tempo polinomial será possível se e somente se $P=NP$.

Trabalhamos na identificação de sub-rotas, *combs* e *multistars* violadas partindo sempre de uma K -árvore mínima $c/2K$ arestas incidentes ao depósito (solução inteira). Como veremos a seguir, a cada iteração do método subgradiente, podemos resolver de maneira exata e em tempo polinomial o problema de identificação de sub-rotas violadas. Questões relativas à implementação e estruturas de dados utilizadas por 3 classes de restrições serão também discutidas neste capítulo.

VI.2 - Geração de Restrições de Eliminação de Sub-rotas (outra abordagem):

Na abordagem alternativa apresentada aqui, não nos restringimos à geração de sub-rotas a partir de sementes (como proposto por Fisher[94.b]). Em nosso caso, a determinação de restrições violadas é realizada buscando sempre informações presentes na solução do problema lagrangeano corrente (K-árvore). Isto difere da proposta apresentada inicialmente por Fisher[94.b], que apresentava um conjunto de partições independentes sem se ater à evolução de cada problema lagrangeano obtido.

Uma alternativa interessante na geração de sub-rotas violadas é particionarmos nosso conjunto de clientes em m componentes conexas (onde m é um valor entre K e $2K$). Para isso, bastará eliminarmos todas as arestas \bar{x}_{0i} da K-árvore mínima (solução do problema lagrangeano) que ligam os clientes ao depósito.

Para cada uma das componentes conexas S_k (onde $k = 1, \dots, m$), construímos a restrição:

$$\mathbf{s}_k = 2r(S_k) - x(\mathbf{d}(S_k)) \leq 0 \quad (1)$$

$$\text{onde } r(S_k) = \lceil d(S_k) / b \rceil \text{ e } x(\mathbf{d}(S_k)) = \sum_{i \in S_k} \sum_{j \in \bar{S}_k} \bar{x}_{ij} \text{ (sendo } \bar{x}_{ij} = 1 \text{ para } i \in S_k \text{ e } j \in \bar{S}_k \text{).}$$

As variáveis binárias \bar{x}_{ij} representam as arestas presentes na solução do problema lagrangeano.

Como estamos interessados apenas nas restrições que violem nossas restrições de capacidade devemos nos preocupar com partições onde temos $\mathbf{s}_k > 0$ na solução do problema lagrangeano (K-árvore). Desta forma, dualizamos apenas um subconjunto das m partições obtidas nesta iteração do método subgradiente! Como consequência do teorema III.2 (capítulo III), apenas as restrições violadas “poderão” proporcionar um incremento estrito do limite inferior.

Para ilustrar a geração de restrições aqui proposta, vejamos o seguinte exemplo onde temos uma 3-árvore mínima com um vértice fixo (depósito) de grau 6 e 3 veículos de capacidade $b=20$. As demandas de cada cliente estão representadas ao lado de cada nó entre parênteses. A demanda da partição S_i ($p/i=1,2,3$) é representada por $d(S_i)$:

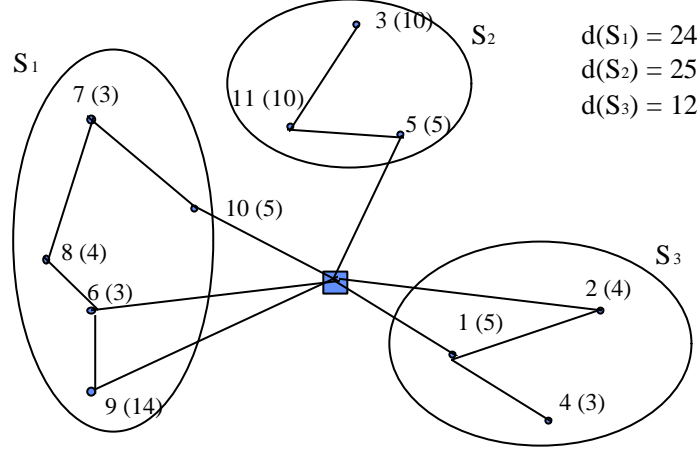


Figura VI.1: Determinação das partições S_i

Note que $r(S_1) = 2$, $r(S_2) = 2$ e $r(S_3) = 1$, são limites inferiores para o número mínimo de veículos que atende a cada subconjunto. Verificando as restrições violadas pela solução do problema lagrangeano para cada uma das partições S_k obtemos:

$$s_1 = 2r(S_1) - \sum_{i \in S_1} \sum_{j \in \bar{S}_1} \bar{x}_{ij} = 4 - 3 = 1 > 0 \quad (\text{rest. violada})$$

$$s_2 = 2r(S_2) - \sum_{i \in S_2} \sum_{j \in \bar{S}_2} \bar{x}_{ij} = 4 - 1 = 3 > 0 \quad (\text{rest. violada})$$

$$s_3 = 2r(S_3) - \sum_{i \in S_3} \sum_{j \in \bar{S}_3} \bar{x}_{ij} = 2 - 2 \leq 0 \quad OK$$

Segue-se que as restrições: $\sum_{i \in S_1} \sum_{j \in \bar{S}_1} x_{ij} \geq 4$ e $\sum_{i \in S_2} \sum_{j \in \bar{S}_2} x_{ij} \geq 4$ devem ser dualizadas e

adicionadas ao nosso conjunto de restrições ativas.

Temos então a seguinte heurística para determinação de sub-rotas violadas:

PROCEDIMENTO VI.1: {Heurística p/ determinação de sub-rotas violadas}

Início

- achou_rv = FALSO; {indica se achamos ou não rest. violadas}
- desconectamos todos os clientes do depósito;
 - {obtemos m componentes conexas S_i (para $i=1,...,m$)}
- calculamos $x(\delta(S_i))$, para $i=1,...,m$;
- **para** ($i=1,...,m$) **faça**
 - calcula $s_i = 2 \left\lceil \frac{d(S_i)}{b} \right\rceil - x(d(S_i))$;
 - **Se** ($s_i > 0$) **então**
 - adiciona partição S_i ao conjunto ativo;
 - achou_rv = VERD;
 - fim**;
- **fim**;
- retorna achou_rv; {retorna falso ou verdadeiro}

fim.

Figura VI.2: Heurística p/ determinação de sub-rotas violadas.

Podemos constatar facilmente que, caso nenhuma restrição violada tenha sido encontrada ao final de nossa heurística, não teremos garantida a existência ou não de sub-rotas violadas. Para ilustrar essa situação, vejamos o exemplo da figura VI.3 onde apresentamos uma 2-árvore com 4 arestas incidentes ao depósito. Ao lado de cada nó estão representadas as demandas de cada cliente:

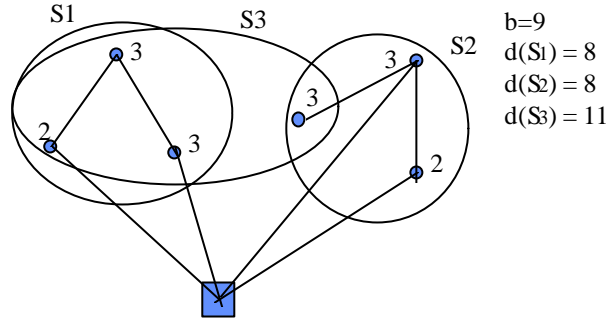


Figura VI.3: Identificação de sub-rotas violadas

Observe que, embora as partições S_1 e S_2 geradas por nossa heurística (componentes conexas) não sejam violadas, a partição S_3 é violada pois:

$$s_3 = 2 \left\lceil \frac{d(S_3)}{b} \right\rceil - x(d(S_3)) = 2 \cdot 2 - 3 = 1 > 0 !!$$

Apresentamos a seguir um algoritmo exato polinomial na identificação de sub-rotas violadas. Este algoritmo retorna VERD ou FALSO conforme existam ou não sub-rotas violadas em nossa K-árvore.

PROCEDIMENTO VI.2: {alg. exato p/ identificação de sub-rotas violadas - versão 1}

Início

achou_rv := procedimento VI.1; {retorna VERD ou FALSO}

Se (não achou_rv) **então**

Se (existir alguma aresta c / um extremo de grau 1) **então**

- adiciona vértices c / grau 1 ao conjunto ativo (restrições dualizadas);

- achou_rv = VERD;

fim;

fim;

retorna achou_rv;

fim.

Figura VI.4: Determinação de sub-rotas violadas (alg. exato).

É fácil ver que cada vértice v com grau 1 em T_k determina uma partição $S_v = \{v\}$ violada $c/\mathbf{s}_v = 1 > 0$ (já que $\left\lceil \frac{d(S_v)}{b} \right\rceil = 1$ e $x(\mathbf{d}(S_v)) = 1$).

Mostraremos formalmente a seguir que o algoritmo VI.2 sempre resolve, de maneira exata, o problema da identificação de sub-rotas violadas em uma iteração qualquer do subgradiente, ou seja, se não existirem vértices com um extremo de grau 1, não existirão sub-rotas violadas. Vejamos inicialmente a seguinte notação:

Notação: Seja S_i (para $i=1,...,m$), cada uma das componentes conexas obtidas desconectando-se os clientes do depósito. Representaremos por $G_i(V_i, E(V_i))$ onde $i=1,...,m$ os subgrafos induzidos por $V_i = S_i \cup \{0\}$. •

Lema VI.1: Se não existirem partições violadas ao final da heurística VI.1, teremos $d(S_i) \leq b$ para $i=1,...,m$.

Prova:

Suponha inicialmente que apenas uma aresta ligue uma componente conexa qualquer S_l ao depósito. Ou seja, $x(\mathbf{d}(S_l)) = 1$ para algum $l \in \{1,...,m\}$. Como $\left\lceil \frac{d(S_l)}{b} \right\rceil \geq 1$, para $i=1,...,m$ temos que $\mathbf{s}_l = 2 \cdot \left\lceil \frac{d(S_l)}{b} \right\rceil - 1 > 0$! O que é absurdo, pois não temos sub-rotas violadas ao final da heurística VI.1. Portanto:

$$x(\mathbf{d}(S_i)) \geq 2 \quad \forall i = 1, \dots, m \quad (I)$$

Suponha (por absurdo) que $d(S_l) > b$ para algum $l \in \{1,...,m\}$. Teremos então $\left\lceil \frac{d(S_l)}{b} \right\rceil \geq 2$. Desta forma, o subgradiente associado a S_l será dado por

$$s_s = 2 \cdot \left\lceil \frac{d(S_l)}{b} \right\rceil - x(d(S_l)) \leq 0 \quad (\text{já que não temos restrições violadas ao final da heurística VI.1}).$$

Ou seja:

$$x(d(S_l)) \geq 4 \quad (II)$$

Calculando o número de arestas para cada subgrafo G_i (como definido acima) e observando que cada componente conexa S_i tem pelo menos $|S_i| - 1$ arestas, temos de (I) e (II) que $|E(V_i)| \geq |S_i| + 1$ para $i=1, \dots, m$ sendo $i \neq l$ e $|E(V_l)| \geq |S_l| + 3$ para $i=l$. Ou seja, o número mínimo de arestas em nosso grafo G , será obtido fazendo-se:

$$\begin{aligned} x(d(S_i)) &= 2 & p / \quad i \neq l \\ x(d(S_l)) &= 4 & p / \quad i = l \end{aligned} \quad (III)$$

É fácil ver que, como temos a componente l ligada ao depósito por 4 arestas, e as demais componentes ligadas ao depósito por 2 arestas, teremos $m=K-1$ componentes conexas obtidas ao final da nossa heurística.

Segue que o número mínimo N_{min} de arestas em G será dado por:

$$N_{min} = |E(V_1)| + \dots + |E(V_l)| + \dots + |E(V_m)| = |S_1| + 1 + \dots + |S_l| + 3 + \dots + |S_m| + 1 \quad (IV)$$

Como $\sum_{i=1}^m |S_i| = n$ e $m=K-1$, temos de (IV) que, $N_{min} = n + m + 2 = n + K + 1$, o que é absurdo pois nossa K árvore contém exatamente $n+K$ arestas. Portanto, $d(S_i) \leq b$ para $i=1, \dots, m$. •

Vejamos agora o seguinte lema:

Lema VI.2: Se não existirem partições violadas após o término da heurística VI.1, teremos exatamente 2 arestas unindo cada componente conexa ao depósito. Além disso, cada uma dessas componentes define uma árvore.

Prova:

Caso não tenhamos partições violadas ao final da heurística VI.1, teremos $d(S_i) \leq b$ para $i=1, \dots, m$ (lema anterior). Segue diretamente que $x(\mathbf{d}(S_i)) \geq 2$, para $i=1, \dots, m$ (já que $s_i = 2 - x(\mathbf{d}(S_i)) \leq 0$ qualquer que seja $i \in \{1, \dots, m\}$).

Suponha agora que para alguma componente S_l (onde $l \in \{1, \dots, m\}$) tenhamos $x(\mathbf{d}(S_l)) \geq 3$. Teremos pelo menos $|S_l| - 1$ arestas em S_l para $i=1, \dots, m$ (já que cada S_i é conexa). Segue que $G_i(V_i, E(V_i))$ (subgrafo induzido por $V_i = S_i \cup \{0\}$), contém pelo menos $|S_l| + 2$ arestas se $i=l$, e $|S_l| + 1$ arestas se $i \neq l$.

Se m componentes conexas forem geradas então:

$$\sum_{i=1}^m |E(V_i)| \geq |S_1| + 1 + \dots + |S_l| + 2 + \dots + |S_m| + 1$$

Como $\sum_{i=1}^m |S_i| = n$ e $\sum_{i=1}^m |E(V_i)| = n + k$, teremos $n + K \geq n + m + 1$. Ou seja $m \leq K - 1$.

Assim, $m \leq K - 1$ componentes conexas foram geradas. Note entretanto, que precisamos no mínimo, K partições (K veículos) para atender toda a demanda de maneira satisfatória. Logo, para alguma componente S_l (onde $l \in \{1, \dots, m\}$), teremos $d(S_l) > b$, o que é absurdo pois $d(S_i) \leq b$ para $i=1, \dots, m$.

Assim, qualquer que seja a componente conexa S_i obtida (para $i=1, \dots, m$) teremos $x(\mathbf{d}(S_i)) = 2$.

De maneira análoga, mostramos que cada componente conexa S_i ($p/ i=1, \dots, m$) contém exatamente $|S_i| - 1$ arestas ao final da heurística VI.1. Suponha $p/$ absurdo, que alguma componente conexa S_l tenha pelo menos $|S_l|$ arestas. Note que, cada componente conexa S_i $p/ i \neq l$, possui pelo menos $|S_l| - 1$ arestas. Como $x(\mathbf{d}(S_i)) = 2$ $p/ i=1, \dots, m$ e $m=k$, obtemos a seguinte desigualdade estrita:

$$\sum_{i=1}^m |E(V_i)| = \sum_{i=1}^m (|E(S_i)| + 2) > \sum_{i=1}^m (|S_i| - 1 + 2) = \sum_{i=1}^m |S_i| + m = n + K$$

Chegamos portanto a um absurdo já que $\sum_{i=1}^m |E(V_i)| = n + K$. Logo, cada componente

conexa S_i define uma árvore. •

Teorema VI.1: Seja T_K o subgrafo gerado a partir da solução do problema lagrangeano. O algoritmo VI.2 sempre determina, caso exista, uma sub-rotas violada.

Prova:

Suponha que nenhuma partição violada S_i seja obtida após o término da heurística VI.1. Mostraremos que, se não existir nenhuma aresta c com uma extremidade com grau 1, então não existirão sub-rotas violadas em nossa K-árvore.

Note que, se ao término do algoritmo VI.2, sub-rotas violadas não forem encontradas, todos os vértices de G terão grau superior ou igual a 2.

Do lema anterior, temos exatamente 2 arestas incidentes ao depósito para cada subgrafo G_i obtido. Como cada componente conexa S_i define uma árvore, teremos a formação de um único ciclo C_i em G_i . Sejam $(0, v_1)$ e $(0, v_2)$ (onde $v_1 \neq v_2$), as arestas que unem S_i ao depósito. Obviamente teremos um único caminho em S_i que une v_1 a v_2 .

Suponha que exista algum vértice neste caminho com grau superior ou igual a 3. É fácil ver neste caso que $\bar{V}_i = V_i \setminus C_i$ é não vazio. Como S_i é uma árvore (lema anterior) e $\bar{V}_i \subset S_i$ segue que $\bar{G}_i(\bar{V}_i, E(\bar{V}_i))$ define uma floresta onde cada árvore possui sua raiz no caminho de v_1 a v_2 . Desta maneira, teremos vértices de V_i com um extremo de grau 1, o que é absurdo pois teríamos encontrado uma sub-rotas violada (algoritmo VI.2).

Portanto, não teremos nenhum vértice (diferente do depósito) no ciclo C_i com grau diferente de 2. Assim, se para cada subgrafo G_i , as restrições de capacidade forem satisfeitas (lema VI.1) e todos os vértices em G_i tiverem grau 2, teremos uma solução viável para o

PRV(problema de roteamento de veículos). Logo, não existirão sub-rotas violadas em nossa K-árvore (teorema IV.2.1). •

Uma nova versão mais sofisticada do algoritmo VI.2 pode ser apresentada. Neste caso, buscamos a geração de um número maior de sub-rotas violadas a cada iteração. Esta idéia busca combinar a heurística VI.1 com a heurística de Fisher (capítulo IV) na geração de sub-rotas violadas. Em sua heurística, Fisher constrói partições (utilizando a idéia de nó semente) sem se ater ao problema lagrangeano corrente. Em nosso caso, buscamos a construção de um “aninhamento” de partições, todas violadas e geradas a partir da K-árvore mínima T_K (solução do problema lagrangeano).

Consideramos inicialmente $D_K \subset T_K$, o conjunto de todas as arestas de T_K não adjacentes ao depósito. Definimos agora $\Phi = \{T \subset D_K / T \text{ é árvore e } x(\mathbf{d}(T)) = 1\}$. Observe que cada um dos subconjuntos T de Φ define uma sub-rota violada. A construção destas árvores é iniciada sempre a partir dos vértices de grau 1 em D_K . Na figura VI.5 apresentamos um exemplo particular da geração de sub-rotas:

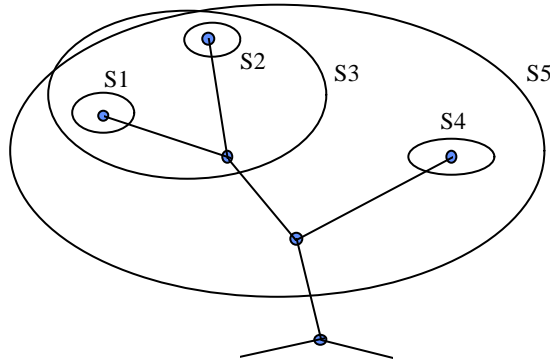


Figura VI.5: Aninhamento de sub-rotas

Note que cada partição S_i como gerada acima determina uma sub-rota violada, já que $x(\mathbf{d}(S_i)) = 1$ e $\left\lceil \frac{d(S_i)}{b} \right\rceil \geq 1$. Desta forma, teremos sempre $s_s = 2 \cdot \left\lceil \frac{d(S_i)}{b} \right\rceil - 1 > 0$!

Chegamos então ao seguinte algoritmo para a geração de sub-rotas “aninhadas”:

PROCEDIMENTO VI.3: {Alg. exato p/ identificação de sub-rotas violadas - versão 2}

Início

- insere restrições violadas (no conjunto ativo) pela heurística VI.1;
- **Enquanto** (existir alguma aresta c / um extremo de grau 1 em D_K) **faça**
 - $D = D_K$;
 - **Enquanto** ($D \neq \emptyset$) **faça**
 - seleciona aresta (a, b) de D ;
 - $D = D - \{(a, b)\}$;
 - **Se** ($\text{grau}(a) = 1$ ou $\text{grau}(b) = 1$ (em D_K)) **então**
 - **Se** ($\text{grau}(a) = 1$) **então**
 - adiciona contração **a** ao conjunto ativo;
 - $\text{grau}(b) = \text{grau}(b) - 1$;
 - senão**
 - adiciona contração **b** ao conjunto ativo;
 - $\text{grau}(a) = \text{grau}(a) - 1$;
 - fim se**;
 - $D_K = D_K - \{(a, b)\}$;
 - fim se**;
- fim enq**;
- fim enq**;
- fim.**

Figura VI.6 : Construção de sub-rotas violadas

Note que, a cada passo do “aninhamento” de sub-rotas fazemos a contração de uma aresta (a,b) em D_K . Se o grau de **a** for 1, adicionamos **a** e todos os vértices associados ao extremo **a** no conjunto ativo (conj. de restrições dualizadas). Em seguida, o extremo **a**

desaparece e todos os vértices associados a **a** serão associados ao extremo **b**. Chegamos ao final do algoritmo quando não existirem mais vértices de grau 1 em D_K .

Observe que o algoritmo VI.3 também resolve, de maneira exata, o problema de identificação de sub-rotas. O “aninhamento” de sub-rotas é obtido sempre a partir de um vértice de grau 1!

Resultados computacionais satisfatórios serão apresentados no capítulo IX, onde mostramos a eficiência do algoritmo exato proposto.

VI.2.1- Implementação e estrutura de dados utilizada na geração de sub-rotas:

Como discutido no capítulo III, devemos estabelecer critérios que permitam atualizar dinamicamente o conjunto das restrições dualizadas.

No caso das restrições de eliminação de sub-rotas, se considerarmos o *lifting* introduzido por Fisher[94.b], temos que $\mathbf{s}_S = 2r(S) - \sum_{j=0}^n e_j \sum_{i \in S} x_{ij} \leq 0$, é a coordenada do vetor de subgradiantes associada à partição $S \subseteq N$, onde $|S| \geq 2$ (capítulo IV). Assim, sempre que $\mathbf{s}_S > 0$, adicionamos a partição S ao conjunto ativo.

Em nossa abordagem, caso o multiplicador \mathbf{n}_S (associado a S) seja nulo, retiramos a partição S do conjunto ativo. Teremos neste caso $\mathbf{s}_S \leq 0$, já que $\mathbf{n}_S = \max\{0, \mathbf{n}_S + \mathbf{q}\mathbf{s}_S\}$ é a formula de atualização dos multiplicadores. Desta forma, a partição S não irá mais contribuir para o incremento estrito do limite inferior (vide capítulo III). Cabe aqui ressaltar que, apenas as restrições ora violadas e um subconjunto das partições (violadas ou não) associadas a multiplicadores de Lagrange não nulos definem nosso conjunto ativo (conjunto de restrições dualizadas). Fisher[94.b], ao contrário, mantém restrições dualizadas (com multiplicadores nulos) por mais 3 iterações do subgradiente.

Como armazenar entretanto as partições associadas às restrições violadas? Além disso, como retirarmos as restrições associadas às partições que não pertencem mais ao conjunto ativo? Observe que, como temos uma atualização dinâmica das partições do conjunto ativo, será

interessante utilizarmos uma estrutura de dados dinâmica que aloque e desaloque memória para estas partições quando necessário.

Para ilustrar a estrutura de dados utilizada vejamos a 2-árvore representada na figura VI.7. Suponha que tenhamos um veículo de capacidade $b=12$ e que as demandas de cada cliente estejam representadas ao lado de cada nó entre parênteses:

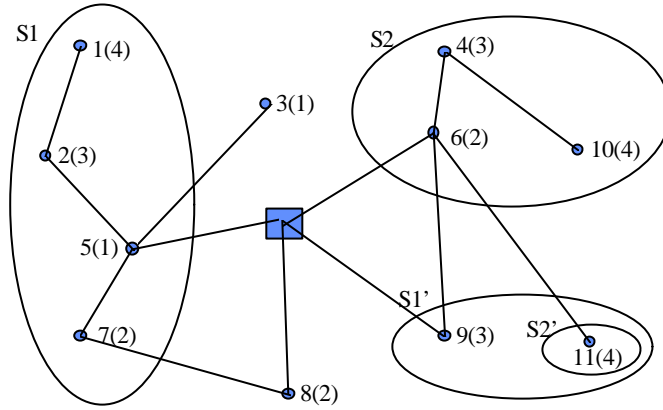


Figura VI.7: Representação das partições S1 e S2

A estrutura utilizada na representação das partições S1 e S2 é ilustrada na figura VI.8. Representamos também os conjuntos S1' e S2', necessários na implementação do *lifting* para as sub-rotas.

Note, na figura IV.8, que para cada partição S_k temos campos associados ao número de elementos $|S_k|$, demanda $d(S_k)$, multiplicadores \mathbf{n}_{S_k} , valor da restrição \mathbf{s}_k , um ponteiro para outra lista (indicando os elementos presentes nos conjuntos S_k e S_k') e finalmente, um ponteiro para a próxima partição S_{k+1} .

Como trabalhamos com o *lifting* introduzido por Fisher[94.b], temos associada a cada partição S_k uma outra partição $S_k' = \{j \in N \setminus S_k / j \geq 1 \text{ e } d_j > br(S_k) - d(S_k)\}$. Assim, associamos o valor 1, a um vértice v se $v \in S_k$, e -1, se $v \in S_k'$. Se v não pertence a nenhum desses dois conjuntos associamos então o valor *zero*.

Note que $\mathbf{s}_2 = 2r(S_2) - \sum_{j=0}^n e_j \sum_{i \in S_2} x_{ij} = 2 * 1 - (x_{69} + x_{06}) = 0$, já que $e_j = 0$ para $j \in S_2'$

e $|S_2'| \leq 2$ (capítulo IV). Observe neste caso que $j=11$ pertence a S_2' e $|S_2'| = 1 \leq 2$, assim,

$$e_{11}x_{6,11} = 0 !!$$

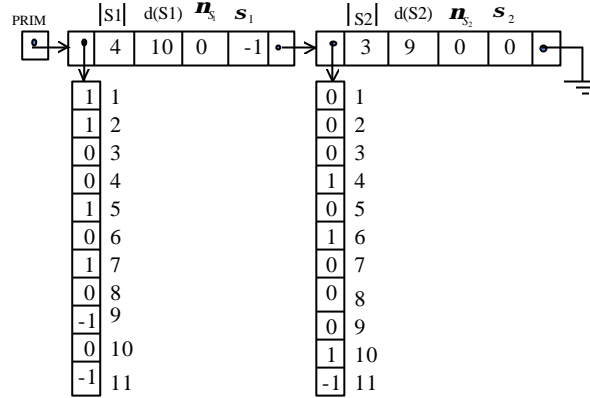


Figura VI.8: Construção das partições S_k

No cálculo do somatório $\sum_{j=0}^n e_j \sum_{i \in S} x_{ij}$, estaremos interessados em saber se uma determinada aresta (a,b) de T_k com um dos extremos em S pertence ao não a $x(\delta(S))$. Desta forma, verificamos se a soma dos valores associados a **a** e **b** (no vetor de partições) é igual a 0 ou 1. Se esta soma é 2, as duas extremidades de (a,b) estão em $S1$ (ou $S2$). Se esta soma é menor ou igual a -1, nenhuma das extremidades de (a,b) pertence a $S1$ ou $S2$ respectivamente. Nestes 2 casos teremos $(a,b) \notin x(\delta(S))$. Caso esta soma seja 0, e **a** (ou **b**) pertença à partição analisada, teremos (a,b) com uma extremidade em $S1$ (ou $S2$) e outra extremidade em $S1'$ (ou $S2'$). Este tipo de representação será importante na implementação do *lifting* proposto por Fisher[94.b].

Antes de adicionarmos uma restrição violada no conjunto ativo, devemos verificar se a partição correspondente já pertence ao conjunto ativo! Ao computarmos o número de elementos e a demanda total de cada partição, desenvolvemos um mecanismo que permite uma checagem rápida de partições já existentes. Note que, se uma partição S_k tem o mesmo número de elementos e a mesma demanda que uma partição S_p , teremos provavelmente duas partições idênticas. Somente neste caso, será necessário fazer uma comparação um a um entre os

elementos (clientes) de S_p e S_k . Assim, dada uma partição S_p (com $\mathbf{s}_{S_p} > 0$), devemos percorrer o conjunto ativo comparando a partição S_p com todas as partições presentes no conjunto ativo (de tamanho pré-determinado M_1). Caso a partição S_p não pertença ao conjunto ativo, adicionamos esta partição na posição apontada por PRIM (vide figura VI.8).

Ao incrementarmos o conjunto ativo, o número de contrações (partições) obtidas a cada iteração do método subgradiente é limitado por \mathbf{n} (vide procedimento VI.3). Como o número de clientes de cada partição também é limitado por \mathbf{n} , teremos uma complexidade total de $O(M_1 n^2)$ iterações no procedimento que adiciona sub-rotas ao conjunto ativo. Lembre-se que antes de inserir uma nova partição ao conjunto ativo devemos compará-la com as partições já existentes.

Observe que, se após a atualização dos multiplicadores (na expressão $\mathbf{n}_{S_k} = \max\{0, \mathbf{n}_{S_k} + \mathbf{q}\mathbf{s}_k\}$) tivermos ainda algum multiplicador nulo, a partição S_k associada deverá ser retirada do conjunto ativo. Para isso, percorremos nossa lista encadeada a partir da posição inicial apontada por PRIM (primeira partição do conjunto ativo).

VI.3 - Identificação de *combs* (penteados) violadas:

De maneira análoga à geração de sub-rotas generalizadas buscamos uma estratégia que identifique *combs* violadas através da solução do problema lagrangeano. Mostraremos que, para uma grande quantidade de estruturas especiais (em nossa K-árvore mínima), conseguimos identificar facilmente a *handle* e dentes (*teeth*) de uma *comb* violada.

Consideramos a situação (apresentada por Cornuejols e Harche [93]) que, ao contrário de Laporte e Nobert[87], não trabalham com as demandas associadas a cada cliente. A utilização da formulação envolvendo demandas, apesar de interessante, desconsidera a presença do depósito. Ao escolhermos a formulação de Cornuejols e Harche [93], optamos por uma expressão que permitisse uma fácil identificação (a partir de T_K) e que redundasse obviamente em um pequeno esforço computacional. Testes computacionais entretanto, devem ser realizados comparando estas duas formulações.

Assim, devemos identificar uma *handle* H e dentes T_1, \dots, T_s de maneira que a seguinte restrição seja violada:

$$x(E(H)) + \sum_{i=1}^s x(E(T_i)) \leq |H| + \sum_{i=1}^s |T_i| - \frac{3s+1}{2} + \alpha(K-1) \quad (2)$$

- onde:
- (i) $|T_i \setminus H| \geq 1$, para $i = 1, \dots, s$;
 - (ii) $|T_i \cap H| \geq 1$, para $i = 1, \dots, s$;
 - (iii) $|T_i \cap T_j| = 0$, para $1 \leq i < j \leq s$;
 - (iv) s é ímpar e $s \geq 3$.

$$\alpha = \begin{cases} 0; & \text{se } 0 \notin H \cup \left(\bigcup_{i=1}^s T_i \right) \\ 1; & \text{se } 0 \in H \setminus \left(\bigcup_{i=1}^s T_i \right) \\ 2; & \text{se } 0 \in H \cap T_j; \text{ para algum } j = 1, \dots, s \end{cases}$$

Trataremos aqui, apenas das situações onde temos $\alpha = 1$ e $\alpha = 0$ respectivamente. Para facilitar a distinção dos 2 casos aqui discutidos, utilizaremos a seguinte definição:

Definição VI.1: Suponha que uma *comb* violada tenha sido construída a partir da solução do problema lagrangeano. Chamaremos esta *comb* de **bigcomb**, se o depósito pertence à *handle* e não pertence a nenhum de seus dentes. Caso o depósito não pertença a *handle* (bem como a nenhum de seus dentes) esta *comb* será chamada **smallcomb**. •

Consideramos ainda a seguinte definição:

Definição VI.2: Dada uma K-árvore T_K . Chamaremos de *folha*, a cada uma das arestas de T_K , não adjacentes ao depósito (vértice 0) e com um extremo de grau *um*. •

Note ainda, a partir da definição VI.2 acima, que 2 folhas serão *disjuntas* se não tiverem um vértice em comum.

Como veremos a seguir, encontramos freqüentemente, estruturas especiais que permitem uma fácil identificação de *combs* violadas, a partir da solução do problema lagrangeano.

Suponha que nossa K-árvore mínima (c/ 2K arestas incidentes ao depósito) contenha pelo menos 3 folhas disjuntas entre si. A seguinte proposição garante a existência de uma *bigcomb* violada em nossa K-árvore mínima.

Teorema VI.2: (Condição suficiente p/ determinação de *combs* violadas)

Seja $G(N_0, T_K)$ o grafo associado a uma solução do problema lagrangeano. Considere ainda Q , o conjunto de todas as folhas de T_K disjuntas entre si. Se tivermos $|Q| \geq 3$, teremos garantida a existência de uma *comb* violada em T_K .

Prova:

Seja $Q \subset T_K$, um subconjunto de arestas como definido acima. Nossa prova consiste em, utilizando o conjunto Q , construir uma *comb* a partir de T_K , e mostrar em seguida que ela é violada pela solução do problema lagrangeano.

Esta *comb* pode ser gerada tomando-se todos os vértices de $G(N_0, T_K)$ com grau superior ou igual a 2, inserindo-os em seguida na *handle* H . Além disso, escolhemos 3 dentes de cardinalidade 2, cada um contendo uma aresta de Q .

Devemos mostrar agora que esta *comb* é realmente violada pela solução do problema lagrangeano. Para isto considere \bar{Q} , com cardinalidade n_i , o conjunto de todas as folhas de $G(N_0, T_K)$ (vide figura VI.8). Note que as folhas de \bar{Q} não são necessariamente disjuntas entre si. Assim, $Q \subseteq \bar{Q}$.

Tomando-se exatamente 3 arestas disjuntas de Q e lembrando que nossa K-árvore possui $n+K$ arestas, temos que $n+K-n_i$, será o número de arestas com ambas as extremidades

pertencentes a *handle* (as demais arestas pertencem a \overline{Q}). Assim, $x(E(H)) = n+K-n_i$ e $|H| = n+1-n_i$.

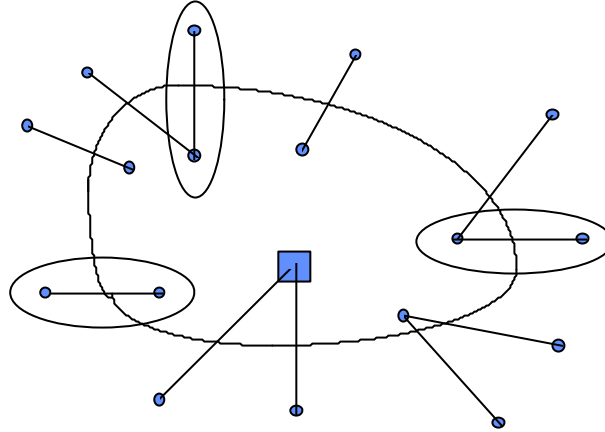


Figura VI.8: Construção de uma bigcomb

Fazendo-se $s=3$, e escolhendo 3 arestas de Q para construção dos dentes temos:

$$\sum_{i=1}^3 x(E(T_i)) = 3 \quad e \quad \sum_{i=1}^s |T_i| = 6$$

Substituindo estes valores na expressão (2) e lembrando que $\alpha=1$, temos:

$$\mathbf{s}_c = x(E(H)) + \sum_{i=1}^s x(E(T_i)) - |H| - \sum_{i=1}^s |T_i| + \frac{3s+1}{2} - \mathbf{a}(K-1) \quad (3)$$

$$\mathbf{s}_c = n+K-n_i+3-n-1+n_i-6+5-K+1 = 2 > 0.$$

Como $\mathbf{s}_c > 0$, temos uma *comb* violada pela solução do problema lagrangeano. •

Note que teremos garantida a existência de uma *comb* violada sempre que 3 folhas disjuntas forem encontradas em T_K . Uma alternativa interessante, será buscarmos um aumento na violação de cada coordenada \mathbf{s}_c do vetor subgradiente. Uma estratégia neste caso, será tentarmos a introdução de dentes grandes com o maior número possível de arestas, aumentando a violação de \mathbf{s}_c . É fácil observar em (3) que, se tomarmos dentes T com cardinalidade $|T|$

(maior ou igual a 3), e um número de arestas maior ou igual a $|T|$ teremos um aumento na violação de s_c . Construímos dentes grandes facilmente analisando-se as componentes conexas obtidas após desconectarmos o depósito dos clientes. Cada componente conexa T_i (onde $i=1,\dots,m$) terá pelo menos $|T_i|-1$ arestas. Caso tenhamos $n_a \geq |T_i|$ (onde n_a é o número de arestas na componente), toda componente conexa analisada poderá ser tomada como um dente grande de nossa *comb*. Se selecionarmos uma *comb* com apenas 3 dentes será interessante escolhermos inicialmente aquelas componentes cuja diferença $n_a - |T_i|$ seja máxima! Ou seja, escolhemos aqueles dentes que geram maior violação.

Resumindo os passos descritos acima, vejamos o seguinte algoritmo onde construímos uma *bigcomb* com 3 dentes, introduzindo, dentes grandes, sempre que possível:

PROCEDIMENTO VI.4: Heurística *Bigcomb*;

Início

- Construímos Q , conjunto de todas as folhas de T_K disjuntas entre si;
- **Se** $|Q| \geq 3$ **então**
 - Adicionamos à *handle* H , todos os vértices de grau maior ou igual a 2;
 - Construímos $P \subset Q$, subconjunto de todas as arestas de Q pertencentes a componentes conexas distintas;
 - **Se** $(|P| \leq 2)$ **então**
 - Construímos 3 dentes T_1, T_2, T_3 respectivamente, selecionando apenas 3 arestas de Q ; { não determinamos dentes grandes }

senão

- Calculamos $n_c = n_a - |S|$ para cada componente conexa S obtida (n_a é o núm. de arestas de cada componente)
- Ordenamos os n_c 's em ordem decrescente de tamanho;
- **Para** $i = 1$ **até** 3 **faça**
 - Selecionamos componentes conexas S a partir do maior n_c ;

```

- Se  $n_c \geq 0$  então
    - a comp. conexa S obtida determina um dente grande;
senão
    - seleciona uma aresta de S e constrói dente pequeno;

fim para;
fim se-senão;
Se (comb gerada não pertence ao conj. ativo) então
    - adiciona comb ao conj. ativo;
fim se;
fim.

```

Figura VI.9: Construção de *bigcombs*

Note no algoritmo VI.4 que teremos a introdução de dentes grandes apenas se tivermos 3 arestas de Q pertencentes a componentes conexas distintas (conjunto P)! Para entendermos melhor a utilização do conjunto P , considere por exemplo (em uma situação extrema) que todas as arestas de Q pertençam a uma mesma componente conexa S . Caso tenhamos $n_a - |S| \geq 0$, toda a componente seria selecionada e teríamos então uma *comb* com apenas um dente, contrariando a condição (iv) apresentada na definição de *comb*. Se $|P| \geq 3$, a condição (iv) nunca é violada. Assim, podemos gerar componentes conexas S (dentes grandes) sempre que $n_a - |S| \geq 0$!

Para tentarmos um incremento maior no limite inferior obtido, seria interessante buscarmos a construção de um número maior de *combs* violadas a cada iteração. Uma alternativa interessante, é gerarmos *combs* adicionais pesquisando-se apenas as componentes conexas obtidas. Seja $D = \{(0, i) \in T_K / i \in N\}$. Vejamos então o seguinte algoritmo:

PROCEDIMENTO VI.5: Heurística VCOMB;

Início

- Constrói (se possível) *bigcomb* em $G(N_0, T_K)$;

- Se (existe *bigcomb* violada) **então**
 - Constrói $G'(N, T_K \setminus D)$; {obtemos m componentes conexas}
 - Constrói (se possível) *smallcombs* (2-matchings) em cada uma das m componentes;
- fim se;**

Fim.

Figura VI.10: Heurística VCOMB

Para ilustrar a geração de *bigcombs* com dentes grandes e *smallcombs*, vejamos o seguinte exemplo onde temos uma 2-árvore com 3 componentes conexas distintas:

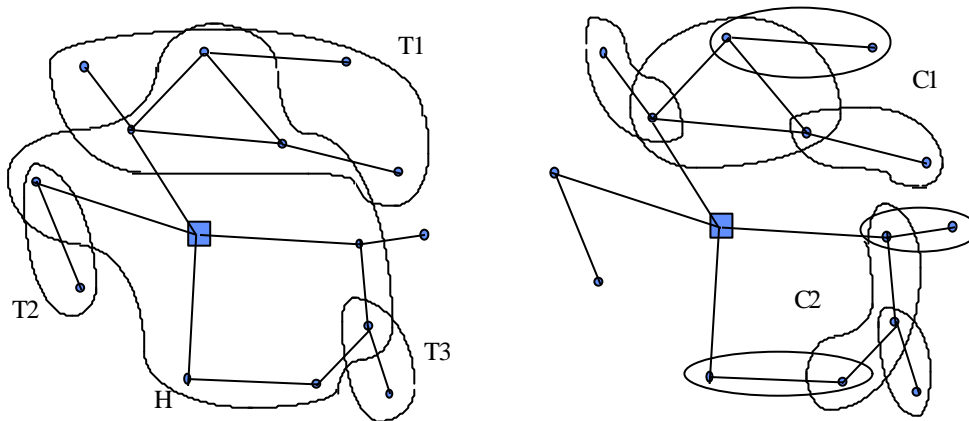


Figura VI.11: Determinação de *bigcombs* e *smallcombs*

A determinação das *smallcombs* é feita de maneira análoga à determinação das *bigcombs*. Para reduzir o esforço computacional entretanto, nos preocupamos apenas com a introdução das 2-matchings (*combs* onde cada dente possui apenas uma aresta) em cada componente conexa gerada. Isto não descarta obviamente, a implementação de outras alternativas.

Apresentamos a seguir o procedimento utilizado na construção das *smallcombs*:

PROCEDIMENTO VI.6: Heurística *Smallcombs*;

Início

- **para** (cada uma das componentes conexas) **faça**
 - construímos Q' , conj. de todas as folhas (da componente) disjuntas entre si;
 - **Se** $|Q'| \geq 3$ **então**
 - Construímos 3 dentes T_1, T_2, T_3 respectivamente, selecionando apenas arestas de Q' ;
 - Adicionamos a H , todos os vért. (da componente) c/ grau maior ou igual a 2;
 - **Se** (*comb* violada não pertence ao conj. ativo) **então**
 - adiciona *comb* ao conjunto ativo;
 - fim;**
- fim;**
- fim.**

Figura VI.12: Construção de *smallcombs*

Note neste caso que, como estamos trabalhando apenas com a identificação de 2-*matchings*, não será necessário a utilização do conjunto $P \subseteq Q$! Resultados computacionais são apresentados no capítulo VIII, avaliando os limites inferiores obtidos após a introdução das desigualdades *comb* (além das restrições de eliminação de sub-rotas) em nosso conjunto ativo.

VI.3.1- Implementação e estrutura de dados utilizada na geração das *combs*:

Fazemos a seguir algumas considerações sobre a estrutura de dados e a implementação realizada na determinação das *combs* violadas: Consideremos o exemplo da figura VI.13 onde temos representada uma 2-árvore com 4 arestas incidentes ao depósito:

Note que, como a componente conexa associada aos vértices $T_1 = \{1, 9, 4, 6, 8, 7\}$ tem 6 arestas (ou seja, $|T_1| - |E(T_1)| \geq 0$) todos os vértices de T_1 serão utilizados na construção de um dente grande. A representação desta *comb* violada é ilustrada na figura VI.14.

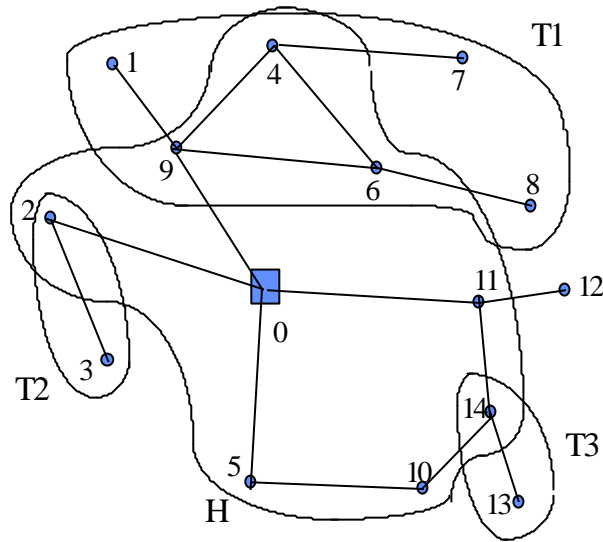


Figura VI.13 : Determinação de *bigcombs*

Temos um vetor de 0's e 1's representando os vértices pertencentes à *handle* H. Um outro vetor é utilizado para a representação dos dentes T_1 , T_2 e T_3 respectivamente. O módulo dos valores negativos indicam o número de elementos presentes em cada dente.

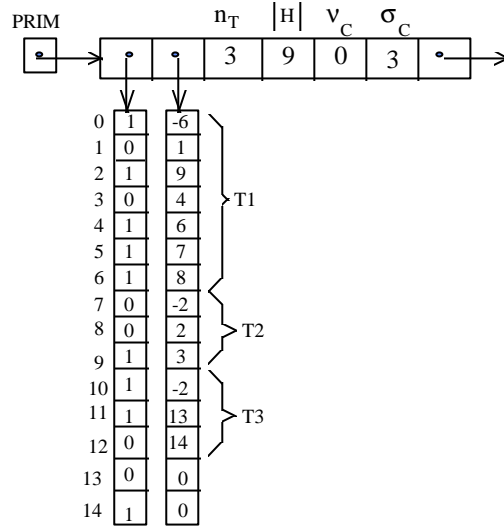


Figura VI.14: Representação do conjunto ativo (*combs*)

Antes de adicionarmos uma *comb* violada ao conjunto de restrições dualizadas (conjunto ativo A), devemos verificar se esta *comb* já pertence a A. Caso a *comb* analisada possua o mesmo número de dentes e a mesma cardinalidade $|H|$ que alguma das *combs* presentes no conjunto ativo, fazemos a comparação um a um dos elementos presentes na *handle e dentes* dessas duas *combs*. Somente após esta checagem, decidimos se a *comb* violada será ou não adicionada ao conjunto ativo.

VI.4 - Identificação de *multistars* (estrelas) violadas:

Trabalharemos agora na identificação das *multistars* apresentadas no capítulo anterior a partir da solução do problema lagrangeano (K-árvore mínima c/ depósito de grau 2K). Assim, devemos determinar subconjuntos S_i (onde $i=1,...,s$) e um vértice \mathbf{v} (núcleo da estrela) de forma que a seguinte restrição seja violada:

$$\sum_{i=1}^s x(\mathbf{d}(S_i)) \geq 4s - 2 \quad (4)$$

- onde:
- i) $S_i \cap S_j = \{v\} \quad \forall i, j \in \{1, \dots, s\} \text{ e } i \neq j$
 - ii) $d(S_i) \leq b; \quad \forall i \in \{1, \dots, s\}$
 - iii) $d(S_i \cup S_j) > b; \quad \forall i, j \in \{1, \dots, s\} \text{ e } i \neq j$

Adicionamos a restrição (4) no conjunto ativo caso $s_s = 4s - 2 - \sum_{i=1}^s x(\mathbf{d}(S_i)) > 0$.

Antes de descrevermos um algoritmo para as *multistars* violadas, vejamos inicialmente algumas definições:

Definição VI.2: Chamaremos de *hélice (ou ponta)* a cada um dos subconjuntos S_i de nossa “estrela”. •

Seja $G(N_0, T_K)$, o grafo associado a nossa K-árvore mínima com depósito de grau $2K$ (solução do problema lagrangeano). Seja ainda $D_K \subset T_K$, o conjunto de todas as arestas de T_K não adjacentes ao depósito. De maneira análoga à geração de sub-rotas violadas construímos $\Phi = \{T \subset D_K / T \text{ é árvore e } x(\mathbf{d}(T)) = 1\}$. Considere ainda, $\overline{\Phi} = \{T \in \Phi / d(T) \leq b\}$. Note que todos os subconjuntos de $\overline{\Phi}$ atendem a condição (ii) acima.

Definição VI.3: Diremos que T_B pertencente a $\overline{\Phi}$ é *hélice base*, se $d(T_B) \geq d(T), \forall T \in \overline{\Phi}$. •

Definição VI.4: O conjunto $\Phi_K = \{T \in \overline{\Phi} / T \not\subset T_B\}$ será chamado *filtro* de Φ . •

Como veremos, se $|\Phi_K| \geq 2$, podemos utilizar cada um dos subconjuntos de Φ_K para a formação das hélices (ou pontas) das *multistars*. A idéia é combinar a hélice base T_B obtida, com as outras sub-árvores T de Φ_K (já filtradas). Se $d(T_B \cup T_i) > b$, (onde $i \neq B$ e $i \in \{1, \dots, |\Phi_K|\}$) a sub-árvore T_i poderá ser utilizada na formação de uma ponta de nossa estrela. Para isto, basta verificar se $d(T_i) + d(\{v\}) \leq b$, onde v (núcleo da estrela), pertence a T_B . Se apenas T_B e T_i forem

geradas, teremos uma estrela de 2 pontas (hélices). Neste caso, $S_1 = T_B$ e $S_2 = T_i \cup \{v\}$ onde $i \in \{1, \dots, |\Phi_K|\}$, $i \neq B$ e $v \in T_B$. A geração de outra ponta pode ser feita de maneira análoga retirando-se T_B e T_i de Φ_K e combinando novamente cada uma das partições T de Φ_K , com as partições T_B e T_i .

A proposição seguinte, estabelece um critério para a escolha do núcleo $v \in T_B$, sempre garantindo a violação das *multistars* como definida anteriormente.

Teorema VI.3: Suponha que as pontas S_1, S_2, \dots, S_s de uma estrela E violada, sejam geradas a partir do filtro Φ_K (onde $s \geq 2$) e satisfaçam as propriedades (i), (ii) e (iii). Se $S_1 = T_B$ e o vértice v (pertencente a T_B), é núcleo de E , então $\text{grau}(v) \leq 3$, $\forall s \geq 2$.

Prova:

Para mostrar nossa afirmação, basta notar que $x(\mathbf{d}(S_1)) = 1$ (já que $S_1 = T_B$) e $x(\mathbf{d}(S_i)) = \text{grau}(v) + 1$, onde $v \in T_B$ e $i=2, \dots, s$. Segue então que:

$$\sum_{i=1}^s x(\mathbf{d}(S_i)) = x(\mathbf{d}(S_1)) + \sum_{i=2}^s x(\mathbf{d}(S_i)) = 1 + (s-1)(\text{grau}(v) + 1)$$

Como a restrição (4) é violada temos $1+(s-1)\text{grau}(v)+(s-1) < 4s-2$. Assim, $\text{grau}(v) < \frac{3s-2}{s-1}$. Podemos mostrar facilmente por indução em s que: $3 < \frac{3s-2}{s-1} \leq 4$, $\forall s \geq 2$.

Como $\text{grau}(v)$ é inteiro e $\text{grau}(v) < \frac{3s-2}{s-1}$, temos que $\text{grau}(v) \leq 3$, qualquer que seja o número de pontas de nossa estrela. •

Para ilustrar a geração de uma estrela com 3 pontas vejamos o exemplo da figura VI.16 onde temos uma 2-árvore com 2 veículos de capacidade $b=32$. Os valores ao lado de cada nó (entre parênteses) representam as demandas de cada cliente:

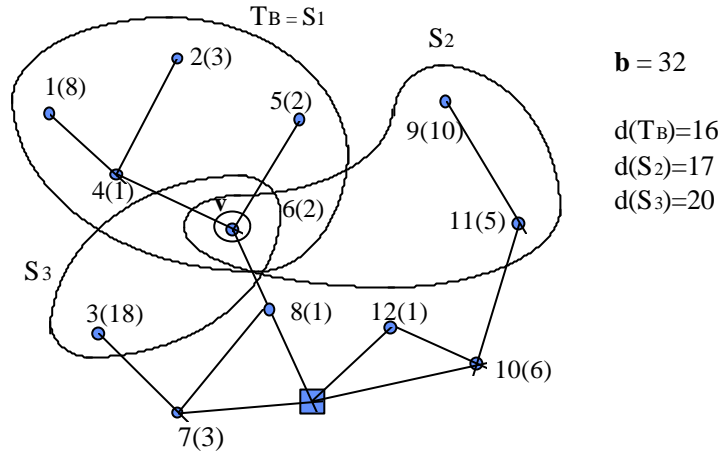


Figura VI.16: Estrela com 3 pontas

Observe que a estrela é violada já que $s_s = 4.3 - 2 - 9 = 1 > 0$!

Resumindo os passos descritos anteriormente, apresentamos o seguinte algoritmo utilizado na geração de estrelas com 2 pontas:

PROCEDIMENTO VI.7: Heurística Estrela2:

Início

- Construímos: $\Phi = \{T \subset D_K / T \text{ é árvore e } x(d(T)) = 1\}$;
- Construímos: $\overline{\Phi} = \{T \in \Phi / d(T) \leq b\}$;
- Seleccionamos $T_B \in \overline{\Phi}$ tal que $\text{demanda}(T_B) \geq \text{demanda}(T_i), \forall T_i \in \overline{\Phi}$;
- Construímos: $\Phi_K = \{T \in \overline{\Phi} / T \not\subset T_B\}$;
- **Para** $i=1$ **até** n **faça**

- **Se** $(\text{grau}(i) \leq 3) \text{ e } (i \in T_B)$ **então** $\{i \text{ é núcleo}\}$

- **Para** (cada uma das sub-árvores T_j de $\Phi_K \setminus T_B$) **faça**

- **Se** $(d(T_B) + d(T_j) > b) \text{ e } (d(T_j) + d(\{v\}) \leq b)$ **então**

- determinamos estrela $E(T_B, T_j, i)$;

Se $(E \notin \text{conj. ativo})$ **faça**

- adiciona estrela $E(T_b, T_j, i)$ ao conjunto ativo;

```

        fim se;
    fim para;
fim se;
fim para;
fim.

```

Figura VI.17 : Geração de *multistars*

Um grande número de estrelas pode ser gerado já que, todos os vértices i de T_B com grau menor ou igual a 3 poderão ser utilizados para a construção de estrelas de 2 pontas. O algoritmo da figura IV.17 pode ser utilizado ainda como base para a geração de estrelas com 3 ou mais pontas.

Como o número de possibilidades (e alternativas) é bastante grande na determinação das *multistars*, optamos por uma solução de baixo custo computacional e que se restringisse apenas na geração de estrelas de 2 pontas, utilizando, como núcleo, apenas vértices de grau 1. Pode-se constatar facilmente, observando a expressão (4), que núcleos de grau 1 permitem a geração de estrelas mais violadas, quando comparadas às estrelas com núcleos de grau 2 e 3 respectivamente.

A determinação de uma estratégia na geração das restrições violadas deve buscar sempre um equilíbrio entre o grau de violação das restrições (e consequentemente a qualidade dos limites inferiores obtidos) e o esforço computacional exigido para obtê-las.

VI.4.1- Implementação e estrutura de dados utilizada na geração das *multistars*:

Apresentamos a seguir a estrutura utilizada na representação das *multistars* violadas.

Vejamos o exemplo da figura VI.16, onde temos uma estrela com 3 pontas. A estrutura de dados associada está representada na figura VI.18:

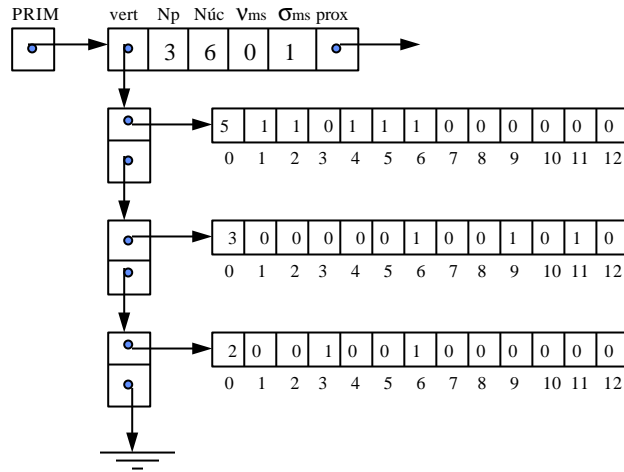


Figura VI.18: Estrutura utilizada na representação de uma *multistar*

Os elementos pertencentes ou não a uma ponta S_k , são representados por 1's e 0's respectivamente. Todas as *multistars* violadas e adicionadas ao conjunto ativo deverão ser inseridas na posição apontada por PRIM. Os valores N_p e Nuc representam, respectivamente, o número de pontas e o núcleo de cada estrela. Os valores n_{ms} e s_{ms} representam o multiplicador de Lagrange e a componente associada ao vetor subgradiente.

Analogamente à inserção de sub-rotas e *combs*, devemos verificar se a *multistar* violada já pertence ao conjunto ativo. Para incrementar a velocidade de busca observamos apenas o número de pontas e o núcleo de cada estrela. Caso 2 estrelas tenham o mesmo número de pontas e o mesmo núcleo comparamos então os demais vértices presentes em cada umas de suas pontas. O número de elementos de cada ponta é representado na posição 0 de cada vetor.

Note que esta estrutura é interessante já que possibilita a geração de uma estrela com um número variável de pontas.