

# Capítulo VIII

## Resolução do problema dual lagrangeano e a busca em árvore

### VIII.1 - Introdução:

Neste capítulo, descrevemos resumidamente, os principais aspectos envolvidos em nossa abordagem na solução do problema de roteamento de veículos com restrições de capacidade. Fazemos uma síntese das principais etapas presentes na solução do problema dual lagrangeano utilizando-se o método subgradiente com geração de desigualdades válidas. Destacamos as etapas de inclusão e eliminação de restrições dualizadas na função lagrangeana, bem como a etapa de fixação de variáveis. Uma nova heurística lagrangeana (seção VIII.3) é também apresentada na determinação dos limites superiores.

Em seguida (seção VIII.4), estudamos a estratégia de ramificação (*branching*) e a busca em árvore (*branch-and-bound*) utilizada em nosso trabalho. Os principais aspectos existentes em nossa implementação são também considerados.

## VIII.2 - Método subgradiente com geração de restrições:

Nesta seção, sintetizamos os principais aspectos envolvidos na resolução do problema dual lagrangeano com a utilização do método subgradiente com geração de restrições. Como discutido anteriormente (capítulos VI e VII), devemos tratar da atualização das restrições dualizadas, composta pelas restrições de eliminação de sub-rotas, *combs* e *multistars*, executando-se, paralelamente, a etapa de fixação de variáveis.

Sejam  $\mathbf{m}, \mathbf{n}_S, \mathbf{n}_C$  e  $\mathbf{n}_M$  os multiplicadores de Lagrange associados às restrições de igualdade, sub-rotas, *combs* e *multistars* respectivamente. Seja ainda  $X$ , o conjunto de todas as  $K$ -árvores mínimas com  $2K$  arestas incidentes ao depósito. Sintetizando as diversas etapas discutidas nos capítulos IV, V e VI, formulamos o problema relaxado (PR) da seguinte forma:

$$z_D = \min \sum_{ij \in E(N_0)} \bar{c}_{ij} x_{ij} + 2 \sum_{i=1}^n \mathbf{m}_i + 2 \sum_{S \subseteq N} v_S r(S) + \sum_{C \subseteq N_0} \mathbf{n}_C r(C) + \sum_{M \subseteq N} \mathbf{n}_M r(M) \quad (\text{PR})$$

$s.a. \quad x \in X$

onde:

$$\bar{c}_{ij} = c_{ij} - \mathbf{m}_i - \mathbf{m}_j - \sum_{j=0}^n e_j \sum_{\substack{i \in S, j \in \bar{S} \\ \text{ou} \\ i \in \bar{S}, j \in S}} v_S + \sum_{\substack{i \in H \\ j \in H}} \mathbf{n}_C + \sum_{p=1}^{n_p} \sum_{\substack{i \in T_p \\ j \in T_p}} \mathbf{n}_C - \sum_{l=1}^{n_m} \sum_{\substack{i \in S_l \\ j \notin S_l}} v_M$$

$$r(S) = \left\lceil \frac{d(S)}{b} \right\rceil$$

$$r(C) = \frac{3n_p + 1}{2} - |H| - \sum_{i=1}^{n_p} |T_i| - \mathbf{a}(k-1)$$

$$r(M) = 4n_m - 2$$

onde os valores  $e_j$  são como definidos no capítulo IV,  $n_p$  é o número de dentes de cada *comb* violada e, finalmente,  $n_m$  o número de pontas de cada *multistar*.

Devemos resolver então o seguinte problema dual lagrangeano:

$$\max z_D(\mathbf{m}, \mathbf{n}_S, \mathbf{n}_C, \mathbf{n}_M)$$

$s.a. \quad \begin{cases} \mathbf{m} \in \mathcal{R}^n \\ \mathbf{n}_S, \mathbf{n}_C, \mathbf{n}_M \geq 0 \end{cases} \quad (\text{PD})$

Na resolução de (PD), utilizamos o método subgradiente (Geoffrion[74], Fisher[81]) combinado à geração de desigualdades válidas.

Seja  $\mathbf{s}_k$ , o valor de cada coordenada (do vetor subgradiente) associado a alguma das restrições dualizadas na função lagrangeana (sub-rotas, *combs* ou *multistars* respectivamente). Como discutido em Geoffrion[74], a atualização dos multiplicadores de Lagrange associado às desigualdades dualizadas ocorre de acordo com a seguinte expressão:

$$I_k = \max \{0, I_k + \theta \mathbf{s}_k\}$$

onde  $I_k$  representa o multiplicador associado à coordenada  $\mathbf{s}_k$ , e  $\theta$  representa o tamanho do passo.

Sejam  $z_{LB}$  e  $z_{UB}$  os limites inferior e superior respectivamente, (obtidos em uma iteração qualquer do subgradiente). O tamanho do passo  $\theta$  será dado por:

$$\theta = P \frac{(z_{UB} - z_{LB})}{\sum_{i=1}^n \pi_i^2 + \sum_{S \in A} \pi_S^2 + \sum_{C \in A} \pi_C^2 + \sum_{M \in A} \pi_M^2} \quad (\text{VIII.1})$$

onde  $\pi$  ( $0 < \pi \leq 2$ ) é decrementado a uma taxa fixa sempre que tivermos um número pré-fixado de iterações do subgradiente sem um incremento estrito do limite inferior.

Note que, se os multiplicadores  $I_k$  associados às restrições já dualizadas são inicializados com 0, eles serão incrementados no passo seguinte já que  $\mathbf{s}_k > 0$  e  $I_k + \theta \mathbf{s}_k > 0$ .

Para cada uma das K-árvores geradas (pertencentes a X) no método subgradiente, devemos atualizar o valor do subgradiente associado àquelas partições já dualizadas. Assim, como discutido no capítulo III, algumas destas restrições poderão ser eliminadas já que não contribuem mais para o incremento estrito do limite inferior. Além disso, a presença destas restrições reduzem desnecessariamente o tamanho do passo no método subgradiente pois, as restrições não violadas (com subgradiente  $\mathbf{s}_k \leq 0$ ) aparecem no denominador da expressão que define  $\theta$ . Sempre que  $\mathbf{s}_k \leq 0$  e  $I_k = 0$ , eliminamos a restrição associada a  $I_k$  de nosso conjunto

de restrições dualizadas. Este procedimento difere daquele desenvolvido por Fisher[94.b], onde são realizadas 3 iterações no subgradiente com o multiplicador nulo antes de sua eliminação!

Como discutido no capítulo anterior, necessitamos de um esforço computacional considerável na implementação do procedimento que fixa variáveis. Além disso, é fundamental que a diferença entre o menor limite superior (*menorzub*) e o maior limite inferior (*maiorzlb*) seja suficientemente pequena para que um maior número de variáveis possam ser fixadas. Executamos então o procedimento que fixa variáveis quando o *gap* de dualidade (diferença entre o limite superior e inferior) for menor que uma porcentagem pré-determinada. Calculamos o *gap* de dualidade (*gapd*) de acordo com a seguinte expressão:

$$gapd = \left( \frac{menorzub - maiorzlb}{maiorzlb} \right) \cdot 100 \quad (VIII.2)$$

Em função do elevado custo computacional exigido pelo procedimento que fixa variáveis, nós o executamos apenas quando o *gap* de dualidade for menor que 7% e o limite inferior for incrementado naquela iteração do subgradiente.

Resumindo os passos descritos acima, apresentamos o procedimento da figura VIII.1 que combina o método subgradiente com a geração de restrições. A constante MAXITER representa o número máximo de iterações permitidas na execução do subgradiente (condição de parada). A constante TOTPASSOS nos indica qual o maior número de iterações do subgradiente sem incremento estrito do limite inferior. Se após TOTPASSOS iterações, o limite inferior não tenha sido atualizado, decrementamos o tamanho de passo *p* a uma taxa pré-determinada TAXAPRED. A constante  $\epsilon$  indica que o subgradiente pode ser executado até que os limites inferiores estejam a uma distancia  $\epsilon$  (suficientemente pequena) do valor da melhor solução viável obtida até o momento (solução  $\epsilon$ -aproximada). Este  $\epsilon$  pode ser modificado conforme trabalhemos ou não com distancias inteiras entre os clientes. No caso de distância inteiras, poderemos utilizar qualquer  $\epsilon \in (0,1)$  para definir o grau de proximidade do valor inteiro  $z_{UB}$ . O mesmo não ocorre para distâncias racionais, situação onde devemos utilizar  $\epsilon$  suficientemente “pequeno”.

A heurística lagrangeana utilizada será discutida com mais detalhes na seção seguinte:

### PROCEDIMENTO VIII.1: Relaxação Lagrangeana c/ Geração de Restrições;

#### Início

$k := 0$ ;

$menorzub := +\infty$ ;                      {inicializa menor limite superior}

$maiorzlb := -\infty$ ;                      {inicializa maior limite inferior}

#### Repita

- Calcula K-árvore mínima c/ 2K arestas incidentes no depósito;
- Calcula limite inferior  $z_{LB}$  ;
- Calcula limite superior  $z_{UB}$  associado a uma solução heurística c/ K rotas;
- **Se**  $z_{UB} < menorzub$  **então**  $menorzub \leftarrow z_{UB}$ ;
- **Se**  $(maiorzlb < z_{LB})$  **então**

$maiorzlb = z_{LB}$ ;

calcula gapd;                      {conforme equação VIII.2}

**Se**  $(gapd \leq 7\%)$  **então**

fixa-variáveis;

iterações := 0;

#### senão

iterações := iterações+1;

**Se**  $(iterações = TOTPASSOS)$  **então**

$p := TAXARED * p$ ;                      {redução no tamanho do passo}

- **fim se**;

- **Se**  $(maiorzlb < (menorzub - \epsilon))$  **então**

- Calcula componentes  $s_k$  do vetor subgradiente associado às restrições já dualizadas;

- Dualiza restrições violadas pela solução do prob. dual lagrangeano ( $s_k > 0$ );

- **Se** (não existirem restrições violadas) **então**

- checa condições de otimalidade;
- **Se** (solução não é ótima) **então**
  - Calcula  $\theta$ ; {conforme equação VIII.1}
  - Atualiza multiplicadores associados às igualdades e desigualdades dualizadas  $\mathbf{S}_k$ ;
  - Elimina todas as partições dualizadas com multiplicadores nulos;
  - Atualiza custos lagrangeanos;
- fim**;
- senão**
  - retornar ao *branch-and-bound*;
- **fim se**;
- Até que** (iterações > MAXITER) ou (tivermos solução ótima);
- fim.** {procedimento}

**Figura VIII.1: Relaxação lagrangeana c/ geração de restrições**

Note que, se restrições violadas não forem encontradas em uma dada iteração do subgradiente, teremos obtido uma solução viável para o problema de roteamento. Para observar esse fato, basta observar se foram encontradas as restrições de eliminação de sub-rotas. Como demonstrado no teorema VI.1, temos um algoritmo exato (algoritmo VI.1) na identificação de sub-rotas violadas. Assim, se não existirem restrições de eliminação de sub-rotas violadas teremos chegado a uma solução viável para o problema de roteamento. Isto não significa entretanto, que tenhamos chegado a uma solução ótima do problema original. Devemos checar ainda se as condições de otimalidade foram ou não satisfeitas, ou seja, devemos verificar se o produto escalar entre o vetor dos multiplicadores de Lagrange e o vetor de subgradientes é igual a *zero* (Geoffrion[74], Fisher[81]).

No capítulo seguinte, mostraremos alguns dos resultados computacionais obtidos em nossa implementação quando atribuímos diferentes valores aos parâmetros do procedimento VIII.1.

### VIII.3 - Clarke&Wright lagrangeano:

Apresentamos nesta seção, uma variação do algoritmo Clarke&Wright (capítuloI/ seção I.2) na geração de limites superiores para o problema de roteamento.

A idéia será utilizarmos o algoritmo Clarke&Wright a cada iteração do subgradiente, definindo portanto, uma heurística lagrangeana na determinação de boas soluções viáveis para o problema original. Para isto, entretanto, dois aspectos principais devem ser observados . Estamos trabalhando com custos positivos e negativos (custos lagrangeanos) e estamos fixando variáveis em 0 e 1 respectivamente.

Seja  $\bar{c}_{ij}$  o custo lagrangeano associado a uma aresta (i,j) qualquer pertencente a E (conjunto de todas as arestas do grafo). Seja ainda  $E(N) \subseteq E$ , o conjunto de todas as arestas de E não adjacentes ao depósito. Representaremos por  $VL_k(N)$ , o subconjunto das arestas de  $E(N)$ , ainda não fixadas em 0 e 1 na k-ésima iteração do subgradiente. As economias (ou *savings*) associadas a estas arestas poderão ser definidas da seguinte maneira:

$$s_{ij} = \bar{c}_{i0} + \bar{c}_{0j} - \bar{c}_{ij} \quad \forall (i, j) \in VL_k(N)$$

Representaremos esta lista de economias (ou *savings*) por  $S_k(N)$ . Diferentemente do algoritmo Clarke&Wright[64], antes de utilizarmos o procedimento na determinação de uma solução heurística, todas as arestas já fixadas em *um* (em uma dada iteração do subgradiente), poderão ser utilizadas na construção de um conjunto de rotas iniciais para o C&W lagrangeano. Na figura VIII.2, representamos um exemplo onde temos as arestas (1,2); (2,3); (0,1) e (6,7) fixadas em *um*:

Devemos buscar agora a formação de novas rotas utilizando-se a lista de economias. Observe que, se fixamos em *um*, alguma aresta (0,i) adjacente ao depósito, poderemos novamente eliminá-la se a economia  $s_{ij}$  (para algum  $j \in N$  e  $i \neq j$ ) for selecionada em nossa lista de economias  $S_k(N)$  durante a execução do C&W lagrangeano. Este fato indesejado pode ser evitado se construirmos um vetor de “graus de liberdade” **GL**, associado aos clientes do grafo.

Antes de executarmos o C&W lagrangeano propriamente dito, inicializamos **GL** fazendo  $GL(i) = 2$  para  $i=1,\dots,n$ . Para todas as arestas  $(i,j)$  já fixadas em 1, decrementamos o “grau de liberdade” associado a **i** e **j** respectivamente. Note por exemplo, que os vértices 5 e 6 na figura VIII.2 tem graus de liberdade iguais a 2 e 1 respectivamente enquanto que o vértice 1 tem grau de liberdade 0. Isto permitirá que a aresta  $(0,1)$  não seja mais eliminada quando estivermos construindo novas rotas através do C&W lagrangeano.

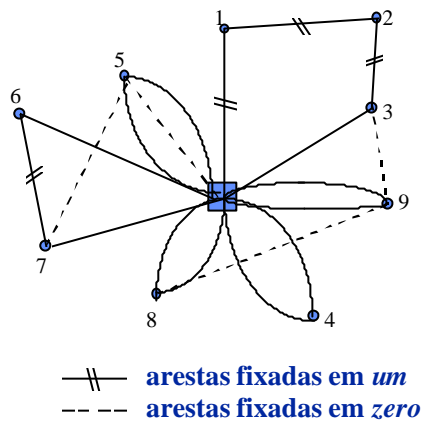


Figura VIII.2: Inicialização no C&W lagrangeano

Observe ainda no exemplo da figura VIII.2 que a aresta  $(0,5)$  (adjacente ao depósito) foi fixada em 0. Isto significa que, embora ela apareça em nossa rota inicial, ela deverá ser eliminada nas iterações seguintes do C&W lagrangeano. Isto poderá ser implementado facilmente atribuindo-se custo lagrangeano *infinito* à aresta  $(0,5)$ .

Depois de construída a rota inicial, construímos novas rotas sempre selecionando as maiores economias de  $S_k(N)$  em ordem decrescente, as economias associadas às arestas já eliminadas (e não adjacentes ao depósito) obviamente não serão analisadas. O processo é repetido até que  $K$  rotas tenham sido geradas (onde  $K$  é o número “mínimo” de rotas viáveis) ou a lista de *savings* esteja vazia.

Note que, ao fazermos a união de duas novas rotas, as economias negativas poderão ser selecionadas piorando o valor da nova solução heurística. Isto pode ocorrer caso uma solução viável com  $K$  rotas ainda não tenha sido obtida. Pode-se gerar instâncias onde uma solução ótima

com  $K+1$  veículos (de mesma capacidade) tenha um valor menor que uma solução ótima com  $K$  veículos!

Antes de apresentarmos as principais etapas presentes em nosso procedimento C&W lagrangeano, vejamos inicialmente a seguinte definição:

**Definição VIII.1:** Chamaremos de *economias viáveis*, representada por  $\bar{S}_k(N)$ , a um subconjunto de economias de  $S_k(N)$  tais que, p/ todas as arestas  $(i,j)$  pertencentes a  $\bar{S}_k(N)$  temos:

- (a) os vértices  $i$  e  $j$  não pertencem a mesma rota;
- (b) a união das capacidades associadas às rotas que contém  $i$  e  $j$  respectivamente não excede a capacidade de cada veículo;
- (c) os vértices  $i$  e  $j$  não são interiores (ou seja,  $i$  e  $j$  estão conectados ao depósito por pelo menos uma aresta).

Temos então o seguinte procedimento, executado na  $k$ -ésima iteração do subgradiente:

**PROCEDIMENTO VIII.2:** Clarke&Wright lagrangeano;

#### **Início**

- Calcula GL a partir das arestas já fixadas em  $um$ ;
- Geramos  $N_r$  rotas iniciais tomando-se as arestas fixadas em  $um$ ;
- Calcula lista de economias  $S_k(N)$ ;
- Ordena (ordem decrescente) a lista  $S_k(N)$ ;

#### **Repita**

- $(a,b) \leftarrow$  aresta de  $S_k(N)$  com maior economia;
- **Se**  $((a,b) \in \bar{S}_k(N))$  **então**
  - **Se**  $(GL(a) > 0)$  e  $(GL(b) > 0)$  **então**
    - Construimos nova rota passando por  $(a,b)$ ;
    - $GL(a) = GL(a) - 1$ ;
    - $GL(b) = GL(b) - 1$ ;

-  $N_r = N_r - 1$ ;

- **fim**;

- **fim**;

reordena lista de economias  $S_k(N)$ ;

**Até que** ( $N_r = K$ ) ou ( $S_k(N) = \emptyset$ );

**fim.**

### Figura VIII.3: Clarke&Wright lagrangeano

Observe que uma aresta (a,b) de  $\bar{S}_k(N)$ , poderá ser rejeitada na formação de uma nova rota se  $GL(a) = 0$  ou  $GL(b) = 0$ . Isto significa que (0,a) ou (0,b) já foram fixadas em *um*, não podendo mais serem descartadas na formação de novas rotas.

Ao final da heurística lagrangeana poderemos ter um número de rotas maior do que K (número de veículos de nossa frota)! Caso isto ocorra, esta solução será descartada e a heurística lagrangeana novamente executada na iteração seguinte do subgradiente (vide procedimento VIII.1).

Se obtemos uma solução viável com K rotas em nosso C&W lagrangeano, fazemos uma busca local utilizando o algoritmo 3-optimal (Lin&Kernighan[73]) em cada uma das K rotas obtidas. Os resultados computacionais obtidos em nossa implementação do C&W lagrangeano serão apresentados no capítulo IX.

## VIII.4 - Busca em árvore (*branch-and-bound*)

Como discutido no capítulo III (seção III.2), sempre que o limite inferior associado a um dado subproblema (ou folha) de uma árvore não corresponder ao valor de uma solução ótima daquele subproblema, particionamos seu conjunto viável associado em dois ou mais subconjuntos. Essa partição (ou *branching*) deverá ser realizada sempre levando em consideração características e informações relativas ao problema original.

Calculados os limites inferiores para cada um dos subproblemas gerados após o *branching* utilizamos uma estratégia de busca (pré-definida), escolhendo um novo sub-problema

a ser pesquisado. Em nosso caso, utilizamos a estratégia do melhor primeiro (ou *best-first*) que consiste basicamente em selecionarmos o subproblema de menor limite inferior entre todos os subproblemas ainda não pesquisados na árvore de busca (vide Murty[74], Horowitz e Sahni[78]). Chamaremos o subproblema selecionado na árvore de busca de **subproblema pai** e os subproblemas obtidos após o *branching* de **subproblemas filhos**.

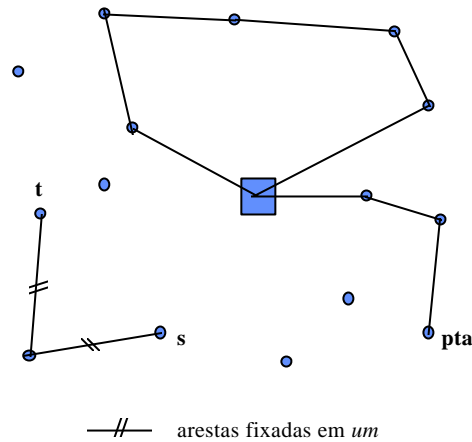
No caso de problemas de minimização, se algum dos limites inferiores associados às folhas ainda não exploradas for superior ao valor da melhor solução viável obtida até o momento (na busca em árvore), o conjunto de soluções viáveis associado a cada uma destas folhas poderá ser descartado definitivamente. Isto permite que folhas sejam eliminadas implicitamente agilizando o processo de busca.

Na seção seguinte discutimos a estratégia de *branching* empregada em nosso trabalho e em seguida os principais aspectos relativos à implementação e estrutura de dados utilizada.

#### VIII.4.1 - Determinação da variável *branching*:

Em nosso trabalho, a árvore de busca é gerada sempre expandindo a folha pesquisada em duas novas folhas (ou subproblemas). Selecionamos uma aresta (variável *branching*) que será fixada em 0 e 1 respectivamente. A escolha desta aresta é realizada sempre a partir da extremidade de alguma rota parcial (ainda não finalizada) que é formada no decorrer do processo da busca em árvore. Ou seja, para determinarmos uma nova variável *branching*, verificamos a extremidade do caminho que tem início no depósito e que ainda não define uma rota completa (vide figura VIII.4). A partir desta extremidade, que chamaremos de **pta**, buscamos o próximo cliente a ser atingido:

Note que, após a utilização do procedimento que fixa variáveis, poderão existir algumas cadeias isoladas de arestas fixadas em *um*. Assim, o próximo nó (a partir de **pta**) deverá ser escolhido de maneira que o grau do próximo cliente seja menor ou igual a *um* e que a demanda associada a este novo percurso não exceda a capacidade de cada veículo. No exemplo da figura VIII.4, se a aresta (pta,s) for escolhida como variável *branching*, o vértice **t** será a extremidade da nova rota (ainda não finalizada) que parte do depósito.



**Figura VIII.4: Determinação de variáveis *branching***

Entre todas as arestas com uma das extremidades em **pta** e candidatas à variável *branching*, escolhemos aquela de menor custo lagrangeano.

Caso a ligação de **pta** com um novo cliente não seja mais possível, tomamos (pta, 0) como próxima variável *branching* (fechando uma nova rota). Se o custo associado a aresta (pta,0) for *infinito*, não será possível a determinação de uma nova variável *branching* a partir de **pta**. Teremos chegado portanto a uma solução inviável.

Outras situações de inviabilidade poderão ocorrer se, ao fecharmos uma nova rota, formarmos uma rota simples, ou ainda, se a demanda associada à rota parcial for inferior a  $\sum_{i \in N} d_i - (K-1).b$ , onde  $d_i$  é a demanda do cliente  $i$ ,  $K$  o número de veículos utilizados e  $b$  a capacidade de cada veículo.

Se estivermos iniciando uma nova rota a partir do depósito, escolhemos o cliente  $i^*$  de maior demanda entre todas aquelas arestas (0,i) incidentes ao depósito ainda não eliminadas.

Escolhida a próxima variável *branching*, fazemos esta variável igual a 0 e 1 respectivamente para determinação dos subproblemas filhos associados à folha pesquisada (subproblema pai). O cálculo destes novos limites inferiores será discutido na próxima seção.

## VIII.4.2 - Implementação e estrutura de dados utilizada

Nesta seção apresentamos os principais aspectos envolvidos em nossa implementação do algoritmo *branch-and-bound* e a estrutura de dados por nós utilizada.

Sempre que um limite inferior associado a um dado subproblema pai, e gerado pelo método subgradiente, não corresponder ao valor de sua solução ótima, devemos gerar dois novos subproblemas filhos (através da variável *branching*) computando em seguida os novos limites inferiores associados a estes subproblemas. Como veremos, o cálculo destes novos limites pode ser feito mais rapidamente aproveitando algumas das informações geradas durante a execução do método subgradiente.

Seja  $\bar{z}_{lb}$  o valor do maior limite inferior associado ao subproblema pai obtido na  $k$ -ésima iteração do subgradiente. Se  $x_{ij}^k$  representa aquelas arestas que estão ou não na K-árvore mínima correspondente temos:

$$\bar{z}_{lb} = \sum_{ij \in E(N_0)} \bar{c}_{ij}^k x_{ij}^k + T_i^k \quad (\text{VIII.3})$$

onde  $\bar{c}_{ij}^k$  é o vetor de custos lagrangeanos gerados na  $k$ -ésima iteração do subgradiente e  $T_i^k$  o somatório dos multiplicadores de Lagrange associados.

Os limites inferiores correspondentes aos subproblemas filhos são gerados através de uma lista de custos lagrangeanos  $\bar{L}$  utilizada na obtenção de  $\bar{z}_{lb}$ . Fixando a variável *branching* em *zero* e *um* respectivamente, geramos duas novas K-árvores mínimas com a utilização de  $\bar{L}$ . O custo de cada uma destas K-árvores somado a  $T_i^k$  nos dá os novos limites inferiores associados aos subproblemas filhos.

Caso um dos novos limites inferiores obtidos tenha um valor menor que a melhor solução viável gerada durante a busca em árvore, adicionamos este limite inferior a uma lista que chamaremos de **lista de abertos** (representada por LA).

Ao escolhermos o subproblema pai na lista de abertos LA, fazemos primeiramente, um pré-processamento fixando novas arestas em *zero* antes da execução do subgradiente! Isto pode

ser feito eliminando-se aquelas arestas ainda não fixadas em *zero* e *um* e que possuam uma extremidade de grau 2. Só após este pré-processamento aplicamos o método subgradiente sobre a lista resultante.

Apresentamos a seguir uma síntese das principais etapas presentes em nossa implementação do algoritmo *branch-and-bound*. A constante *menorzub* abaixo representa o valor da menor solução viável obtida durante a busca em árvore.

#### **PROCEDIMENTO:** *Branch-and-Bound*;

##### **Início**

- Calcula limite inferior (nó raiz);
- **Se** (solução não é ótima) **então**
  - faz pré-processamento tomando-se as arestas fixadas em *um*;
  - calcula var. *branching*;
  - gera 2 novas folhas e calcula limites inferiores associados;
  - **Se** (se esses limites forem menores ou iguais a *menorzub*) **então**
    - incrementa lista de abertos LA;
- **fim**;
- **Enquanto** (  $LA \neq \emptyset$  ) **faça**
  - seleciona folha em LA (conforme estratégia *best-first*);
  - gera lista de arestas associada ao subproblema pai e faz pré-processamento;
  - calcula limite inferior (subproblema pai);
  - **Se** (solução não é ótima) **então**
    - encontra variável *branching*;
    - gera 2 novas folhas (subpb. filhos) e calcula limites inferiores associados;
    - **Se** (se esses limites forem menores ou iguais a *menorzub*) **então**
      - incrementa lista de abertos LA;

**senão**

atualiza *menorzub*;

- fim;

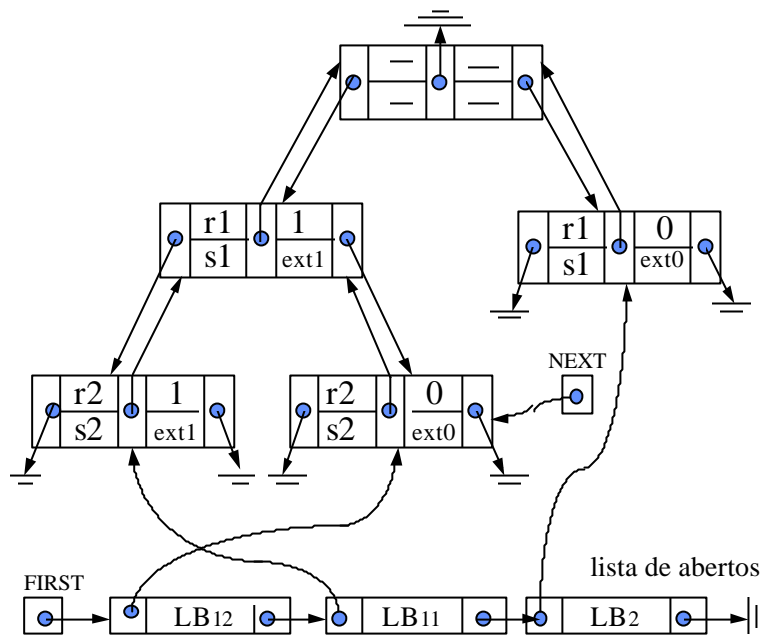
### Figura VIII.5: Algoritmo *Branch&Bound*

Se após o término do subgradiente uma solução ótima não tiver sido obtida, geramos 2 novas folhas de acordo com a estratégia de *branching* utilizada e calculamos os novos limites inferiores associados salvando-se a lista de custos lagrangeanos correspondente ao maior limite inferior obtido durante o subgradiente. Como discutido anteriormente, esta lista será utilizada na determinação dos limites inferiores associados aos subproblemas filhos. Isto é feito, com a utilização da expressão VIII.3.

Vejamos agora alguns aspectos relativos à estrutura de dados utilizada. Determinado o subproblema pai a ser pesquisado, devemos recuperar primeiramente a lista de arestas associada a este subproblema. Isto pode ser feito tomando-se as variáveis *branching* no caminho da folha (associada ao subproblema pai) até a raiz da árvore de busca. Para isto, tomamos primeiramente a lista de arestas correspondente à raiz da árvore de busca. Cada vez que uma variável *branching* for percorrida trocamos a posição desta aresta com a primeira ou última aresta entre aquelas ainda não fixadas. Procedendo desta forma, geramos a lista associada a folha a ser pesquisada apenas quando o respectivo subproblema for selecionado na lista de abertos LA (vide figura VIII.6).

Os valores  $r_i$  e  $s_i$ , em cada folha, representam a última variável *branching* selecionada. Os valores *um* ou *zero*, indicam se a variável *branching* associada foi ou não utilizada. Finalmente, os valores *ext1* e *ext0* indicam a extremidade da rota parcial gerada no decorrer do processo de busca.

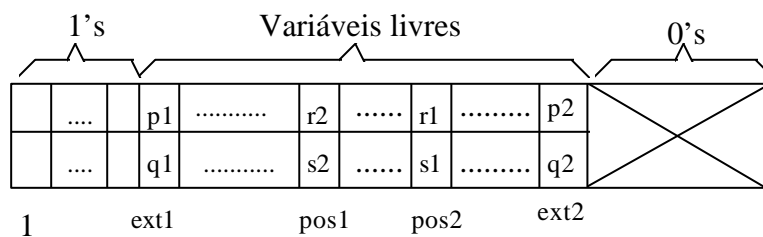
Como discutido anteriormente, ao utilizarmos uma estratégia de busca do tipo *best-first* selecionamos sempre uma folha (indicada por NEXT) cujo limite inferior associado seja o menor entre todos os limites inferiores na lista de abertos (apontada por FIRST).



**Figura VIII.6: Estruturas de dados p/ busca em árvore**

Se LR é a lista de arestas associada à raiz de nossa árvore. Ao gerarmos a lista associada a folha apontada por NEXT, devemos percorrer o caminho da folha até a raiz tomando-se todas as variáveis *branching* presentes neste caminho.

Observando o exemplo da figura VIII.7, note que, ao gerarmos a lista associada ao limite  $LB_{12}$ , devemos trocar as arestas (r2, s2) com (p2,q2) e (r1, s1) com (p1, q1) respectivamente. Em seguida, decrementamos ext2 e incrementamos ext1:



**Figura VIII.7: Lista LR (associada à raiz)**

Este tipo de abordagem irá permitir uma grande economia de memória já que não geramos explicitamente, a lista de arestas associadas à cada folha da árvore.