MÉTODOS DE RESOLUÇÃO PARA O PROBLEMA DE PERCURSOS DE VEÍCULOS

RENATO LOURES BUENO FILHO

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Nelson Maculan Filho, D.Sc.
(Presidente)

Prof. Celso da Cruz Carneiro Ribeiro, D.ING.

Prof[a] Dina Feigenbaum Cleiman, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

ABRIL, 1989.

BUENO FILHO, RENATO LOURES

Métodos de Resolução para o Problema de Percursos de Veículos [Rio de Janeiro] 1989.

ix, 142 p. 29,5 cm (COPPE/UFRJ, M. Sc., Engenharia de Sistemas e Computação, 1989).

Tese - Universidade Federal do Rio de Janeiro, COPPE.

1. Otimização de Percursos de Veículos I. COPPE/UFRJ II. Título (série).

A meus pais, Regina e Renato.

Agradecimentos

A Nelson Maculan Filho, pela orientação e apoio.

A Capes e CNPq, pela ajuda financeira.

A Celso Carneiro Ribeiro e Dina Feigenbaum Cleiman, pela participação na banca examinadora.

A Leu, pelo carinho e apoio incondicional durante todo o curso de mestrado.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para obtenção do grau de Mestre em Ciências (M. Sc.).

MÉTODOS DE RESOLUÇÃO PARA O PROBLEMA DE PERCURSOS DE VEÍCULOS

RENATO LOURES BUENO FILHO

ABRIL, 1989

Orientador: Prof. Nelson Maculan Filho.
Programa: Engenharia de Sistema e Computação.

Neste trabalho apresentamos um estudo dos métodos de resolução para o Problema de Percursos de Veículos (PPV). Inicialmente, discutimos os principais métodos exatos e aproximados de resolução para o PPV básico. Em seguida, apresentamos uma implementação em microcomputador de dois algoritmos aproximados que solucionam o PPV com restrições adicionais relativas aos horários de atendimento dos clientes. Resultados computacionais são apresentados.

Abstract of Thesis presented to COPPE/UFRJ as partial fulfillment of the requirements for the degree of Master of Science (M. Sc.).

METHODS OF SOLUTION FOR THE

VEHICLE ROUTING PROBLEM

RENATO LOURES BUENO FILHO

APRIL, 1989

Thesis Supervisor: Prof. Nelson Maculan Filho.

Department: System and Computer Engeneering.

We present in this work a study of solution methods for the Vehicle Routing Problem (VRP). In its first part, we consider the main exact and approximate methods for the basic VRP. In its second part, we present a microcomputer implementation of two approximate algorithms for the VRP with Time Windows constraints. Computational results are presented.

Índice

CAPÍTULO I

INTRODUÇÃO

O Problema de Percursos de Veículos (PPV) consiste em determinar percursos ótimos para uma frota de veículos estacionada em um domicílio (garagem ou depósito) de forma a atender um conjunto de clientes geograficamente dispersos.

Um exemplo clássico aparece nos problemas de distribuição de mercadorias (DANTZIG & RAMSER, 1959), onde cada cliente possui uma demanda específica e os veículos apresentam uma capacidade limitada. Porém, o problema também aparece em várias situações práticas não relacionadas com distribuição de produtos, como por exemplo, na entrega e/ou coleta de correspondências pelo correio, no transporte de crianças em ônibus escolares, na coleta de lixo, no planejamento de rotas aéreas, etc..

Os problemas de percursos de veículos são na sua grande maioria NP-árduos (LENSTRA & RINNOY KAN, 1981), portanto, pouco prováveis de serem resolvidos em tempo polinomial. O nosso interesse está em desenvolver um sistema para microcomputador que produza soluções de qualidade em um tempo computacional viável. Acreditamos que

isto é possível através da combinação de três fatores: uma modelagem cuidadosa do problema, a utilização de um método de solução eficiente e uma interface adequada com o usuário. Neste trabalho procuramos cumprir uma primeira etapa para o desenvolvimento deste sistema que consiste no estudo dos modelos e métodos de solução para o problema e a implementação de alguns destes métodos em um microcomputador.

Organizamos este trabalho em cinco capítulos. Ainda neste capítulo introdutório, apresentamos uma classificação para os problemas de percursos de veículos e definimos um problema básico. Este problema possui características que normalmente ocorrem nos vários problemas de percursos de veículos encontrados na prática e na literatura e geralmente os métodos que o solucionam podem ser facilmente estendidos ou adaptados para resolver problemas mais complexos. No capítulo II desenvolvemos os principais modelos e métodos exatos de solução e no capítulo III os métodos aproximados. No capítulo IV implementamos alguns destes métodos para resolver um problema de percursos de veículos um pouco mais complexo que aparece muito na vida real. Este problema, que considera restrições adicionais relativas aos horários de atendimento dos clientes, é conhecido na literatura como Problema de Percursos Veículos com restrições de Janela de Tempo. Finalmente, no capítulo V apresentamos algumas conclusões e indicamos futuros trabalhos e pesquisa que podem dar continuidade a este estudo.

I.1 - Classificação e Definição do PPV-básico

Devido a enorme variedade de problemas de percursos de veículos extraídos da vida real convém inicialmente classificá-los em três grandes grupos, a saber, Problemas de Percursos de Veículos (**PPV**), Problemas de Sequenciamento de Veículos (**PSV**) e Problemas de Percursos e Sequenciamento de Veículos (**PPSV**). Uma rota ou percurso de veículo é uma seqüência de pontos de entrega e/ou coleta que o veículo deve percorrer ordenadamente, iniciando e finalizando num depósito ou garagem; portanto, o **PPV** consiste em determinar as rotas que minimizam uma função objetivo, que normalmente é a distância percorrida pela frota. Um sequenciamento de veículos é uma rota onde associamos a cada ponto um tempo de chegada e um tempo de partida, ou de início e fim de serviço, portanto no **PSV** os veículos devem percorrer os pontos numa determinada ordem respeitando os tempos especificados. Quando os aspectos espacial e temporal se combinam de tal forma que um não prevalece sobre o outro temos o **PPSV**. Normalmente esta classe de problemas é caracterizada pela existência de um dos dois (ou ambos) seguintes tipos de restrição, relação de precedência entre dois clientes, isto é, quando é exigido que um determinado cliente seja servido antes que outro e Janela de Tempo ("Time Window") que ocorre quando os clientes exigem que o atendimento seja realizado dentro de um intervalo de tempo.

Naturalmente, esta classificação não é suficiente para definir completamente um problema de percursos de

veículos. Vários problemas podem ser encontrados dentro de cada uma das três classes definidas acima. BODIN & GOLDEN (1981) apresentam uma classificação para os problemas de percursos de veículos baseada nas características dos principais componentes sempre presentes nestes problemas, a saber: o depósito, os clientes, a frota de veículos e a rede. Por exemplo, o **PPV** que vamos definir em seguida possui um único depósito, os clientes possuem uma demanda conhecida, a frota de veículos é homogênea e a rede viária é não direcionada.

Vamos agora definir o **PPV**-básico para nos próximos capítulos apresentarmos os vários métodos de solução.

Considere um grafo $G = (N,A)$ onde $N = \{i = 1,\ldots,n\}$ é o conjunto de nós que representam os clientes, ou pontos de demanda, quando $i \neq 1$ e o depósito quando $i = 1$. Sejam $N' = N - \{1\}$ e $A = \{(i,j): i,j \in N\}$ o conjunto de arcos, cada um possuindo um custo c_{ij} que pode ser interpretado como a distância entre i e j ou como o tempo de viagem entre i e j. Cada cliente possui uma demanda d_i e a frota é composta por m veículos idênticos que possuem capacidade D.

O **PPV**-básico consiste em construir as rotas de custo mínimo de forma que:

(i) Cada cliente de N' deve ser visitado exatamente uma vez de forma a satisfazer sua demanda;

(ii) Cada veículo inicia e termina sua jornada no depósito;

(iii) A demanda total de cada rota de veículo deve ser menor ou igual que a capacidade do veículo, ou seja, $\sum_{i \in R} d_i \leq D$, onde $R \subset N'$;

(iv) O custo total (distância percorrida ou tempo de viajem) de cada rota deve ser menor ou igual que um limite dado L, i.é., $\sum_{i,j \in R} c_{ij} \leq L$.

Podemos classificar os problemas de percursos de veículos de acordo com sua matriz de custos $C = (c_{ij})$. O PPV é dito **Simétrico** quando a matriz C é simétrica, caso contrário o problema é **Assimétrico**. Além disto, o problema é dito **Euclidiano** quando C satisfaz à desigualdade triangular, ou seja, se $c_{ij} \leq c_{ik} + c_{kj}$ $(i,j,k \in N)$. Caso contrário o problema é **Não-euclidiano**.

Geralmente os algoritmos são desenvolvidos para resolver os problemas simétricos-euclidianos, porém são facilmente adaptáveis para resolver os demais casos.

I.2) Classificação dos métodos de solução

Os métodos de solução podem ser divididos em duas grandes classes: os métodos Exatos e os métodos

Aproximados.

Os métodos Exatos fornecem uma soluçao ótima para o problema e baseiam-se em técnicas conhecidas de Programação Matemática, porém sua aplicação é restrita a problemas de pequeno porte, já que a maioria dos problemas de percursos são NP-Árduos (LENSTRA & RINNOOY KAN, 1981].

Um algoritmo Aproximado é um procedimento que utiliza a estrutura do problema em questão aliado a uma heurística, muitas vezes intuitiva, para fornecer uma solução viável "quase-ótima" para o problema. Por solução viável "quase-ótima" entendemos uma solução que satisfaça as restrições do problema e forneça um valor para a função objetivo suficientemente bom ou próximo do valor ótimo. Naturalmente, uma solução aproximada pode ser comparada com um limite inferior para uma formulação exata, como por exemplo o dual de uma relaxação lagrangeana, para a verificação da sua qualidade.

CAPÍTULO II

MÉTODOS EXATOS

II.1 - Introdução

Neste capítulo apresentamos os principais algoritmos exatos para o PPV-básico, mencionamos também algumas extensões importantes quando algoritmos exatos foram desenvolvidos com sucesso.

Os métodos exatos de solução de problemas de percursos de veículos normalmente incluem procedimentos de enumeração do tipo "**Branch & Bound**". Estes métodos estão intimamente relacionados com o modelo utilizado na formulação do problema, portanto, convém classificá-los de acordo com cada modelo. Utilizamos a seguinte classificação para os métodos de solução, sugerida em MAGNANTI (1981) e também seguida por LAPORTE & NOBERT (1985):

(i) Algoritmos de Busca Direta;
(ii) Algoritmos de Programação Dinâmica;
(iii) Algoritmos de Programação Inteira.

Estes últimos podem ser subdivididos em:

a) formulação por Particionamento de Conjuntos;
b) formulação por Fluxo de Veículos;

II.2 - Algoritmos de Busca Direta

Os métodos de busca direta consistem em construir rotas de veículos por meio de árvores de **"Branch & Bound"** (B-B). Métodos enumerativos do tipo B-B solucionam problemas de otimização discreta através do particionamento ("Branch") do conjunto das soluções viáveis em sucessivos subconjuntos de menor dimensão, calculando limitantes ("Bounds") sobre o valor da função objetivo em cada subconjunto. Estes "bounds" são usados para selecionar subconjuntos promissores e/ou para descartar outros que não têm possibilidade de conter a solução ótima. Os "bounds" são obtidos pela substituição do problema original por um problema ***relaxado***, mais fácil de ser resolvido, de tal forma que o valor da solução deste limita o valor da solução do primeiro. O procedimento termina quando cada subconjunto ou produziu uma solução viável ou demonstrou-se que não contém nenhuma solução melhor que aquela obtida até o momento. A melhor solução encontrada durante o processo é então uma solução ótima global.

Os ingredientes essenciais de qualquer algoritmo B-B para resolver um problema P de otimização discreta da forma Min $\{f(x) / x \in S\}$ são:

(i) **Uma relaxação de P**: um problema R da forma Min $\{g(x)/ x \in T\}$, tal que $S \subseteq T$ e para cada $x,y \in S$, $f(x) < f(y) \Rightarrow g(x) < g(y)$;

(ii) **Uma estratégia de "Branching" (separação):** uma regra que particione o conjunto viável S_i do problema corrente P_i em subconjuntos $S_{i_1}, \ldots, S_{i_k}$ tal que $\bigcup_{j=1,k} S_{i_j} = S_i$;

(iii) Uma estratégia de "Lower Bounding": um procedimento para determinar uma solução ótima (ou quase ótima) para cada relaxação R_i de cada subproblema P_i ;

(iv) Uma estratégia de Busca: uma regra para escolher o próximo subproblema a ser processado.

A seguir apresentamos duas estratégias de **"branching"**, que combinadas com diferentes estratégias de **"bounding"** produzem vários algoritmos.

II.2.1 - Branching nos Arcos

Nesta seção apresentamos uma estratégia de "branching" que foi definida por LITTLE et al. (1963) para resolver o Problema do Caixeiro Viajante (PCV) e mostramos como utilizá-la no caso do PPV.

Esta estratégia basea-se na escolha de um **arco** do grafo G para realizar a partição de um subproblema, representado por um nó na **"árvore de busca"** (figura II.1), em dois outros: o que contém todas as rotas que incluem o arco escolhido, e aquele que não inclue este arco nas suas rotas.

Seja x_{ij} a variável de decisão que possui os seguintes valores:

$$x_{ij} = \begin{cases} 1 & \text{se } (i,j) \text{ pertence à solução do problema;} \\ \emptyset & \text{caso contrário.} \end{cases}$$

Portanto cada partição definida acima corresponde à escolha de uma variável x_{ij} (variável de "branching") que se atribui o valor 1 ou Ø. Na figura II.1 abaixo pode-se visualizar este esquema.

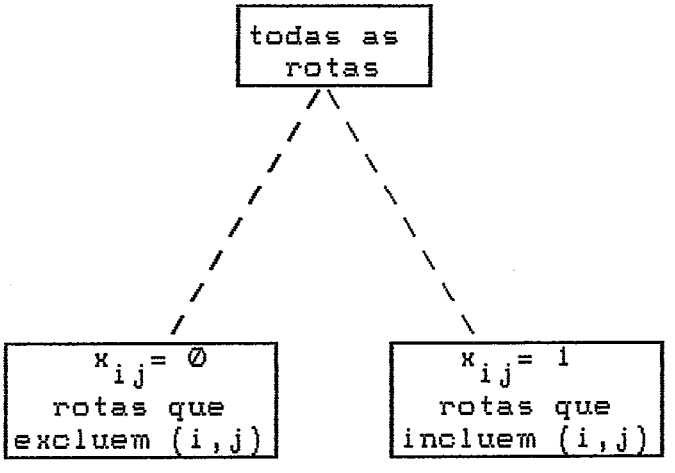


figura II.1 Árvore de Busca

O procedimento de "**branching**" é repetido até que algum nó da árvore de busca acima represente uma solução com rota única ótima.

Convém observar que em qualquer estágio deste processo, a união dos conjuntos representados pelos nós terminais da árvore é o conjunto de todas as rotas do problema.

No algoritmo de LITTLE et al. (1963) uma regra para a escolha da variável de branching é descrita da seguinte maneira.

> "Sejam X, Y e Y' nós da árvore de busca, onde X será o nó a ser particionado nos nós Y, quando $x_{kl} = 1$, e Y' quando $x_{kl} = \emptyset$, onde x_{kl} é a variável de branching escolhida.
>
> Inicialmente, a partir da matriz de custos determine a matriz de "**custos reduzidos**", isto é, uma matriz com elementos não negativos e com pelo menos um zero em cada coluna e linha. Esta matriz é obtida pela seguinte técnica de redução de linhas e colunas: subtrair cada linha (coluna) pelo menor elemento da linha (coluna). Note que os custos relativos de todas as rotas permanecem os mesmos, portanto, qualquer rota ótima na matriz original continua ótima na matriz reduzida.

O próximo passo é efetuar a **estratégia de "branching"**. Esta estratégia basea-se na escolha de um par de clientes de X, que será incluido em Y e excluído de Y', visando construir em Y a melhor rota contida em X. As rotas de menor custo a serem consideradas para Y são aquelas envolvendo um par de clientes (i,j) de Custo Reduzido $c_{ij} = \emptyset$.

Considere agora os custos das rotas que não contêm (i,j), i.é., rotas possíveis para Y'. Como o cliente i deve pertencer à rota do caixeiro viajante, os custos destas rotas devem pelo menos incluir o custo de menor valor entre os custos da linha i, excluindo-se c_{ij}. O mesmo ocorre com a cidade j, portanto, as rotas de Y' também devem incluir pelo menos o custo de menor valor entre os custos da coluna j, excluindo-se c_{ij}. Se definirmos como θ_{ij} a soma destes dois custos, o critério de escolha do par de cidades para o "branching" (variável de branching) será dado por:

$$\theta_{kl} = \text{máx} \{ \theta_{ij} \}.$$

Isto significa uma busca, na matriz reduzida, sobre os (i,j) que possuem $c_{ij} = \emptyset$ e $\theta_{ij} \neq \emptyset$, como mostra a figura II.2."

	1	2	3	4
1	∞	11	27	Ø (11)
2	1	∞	15	Ø (1)
3	Ø (0)	Ø (4)	∞	3
4	Ø (0)	4	Ø (15)	∞

Matriz de custos reduzidos. Os Nos. no canto superior direito são os valores de θ_{ij}.

$\theta_{k1} = \theta_{43} = 15$ portanto (4,3) será o par de cidades escolhido para o branching, e x_{43} a variável de branching.

figura II.2 - Exemplo de escolha de uma variável de "branching".

CHRISTOFIDES & EILON (1969) desenvolveram um algoritmo de B-B baseado na estratégia de "branching" descrita acima para resolver o PPV-básico.

O primeiro passo deste algoritmo consiste em transformar o PPV-básico em um Problema do Caixeiro Viajante (PCV). Isto pode ser feito da seguinte maneira:

> substituir o único depósito por m depósitos artificiais, todos localizados na mesma posição. Viagens entre os depósitos são proibidas atribuindo-se valores iguais a λ para as distâncias entre os depósitos, onde o valor que λ assume depende da F.O. do problema considerado (LENSTRA & RINNOOY KAN, 1975):

$\lambda = \infty$ fornece a distância mínima para m veículos;

$\lambda = \emptyset$ fornece a distância mínima para no máximo m veículos;

$\lambda = -\infty$ fornece a distância mínima para o número mínimo de veículos.

O número de depósitos artificiais m é igual ao número de veículos utilizados na solução ótima. A estrutura da matriz de distâncias do problema modificado, agora um PCV, é mostrada na figura II.3.

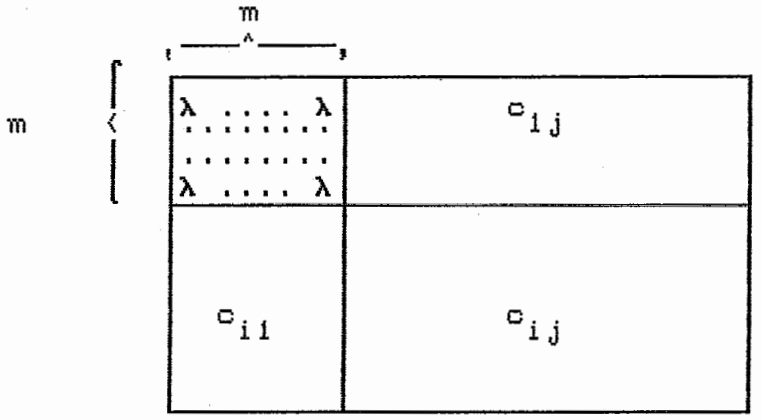


figura II.3 - Matriz de Distâncias para o PPV na forma do PCV.

Para resolver o PPV-básico o algoritmo abandona um "branch" quando :

(i) A carga total para um veículo excede sua capacidade;

(ii) A distância total acumulada por um veículo

excede o limite L;

(iii) A demanda total das cidades ainda não incluídas na rota excede a capacidade total dos veículos ainda não utilizados.

Além das questões acima, que descrevem as restrições de capacidade e distância, é necessáro determinar os "bounds" para os nós da árvore de B-B de maneira a reduzir a busca envolvida (estratégia de "bounding").

Já vimos que uma estratégia para o cálculo de "bounds" baseia-se numa relaxação do problema original. Para melhor visualizarmos as possíveis relaxações para o PCV , consideramos a seguinte formulação do problema em termos de Programação Linear Inteira (LAPORTE & NOBERT, 1985):

$$\text{(PCV)} \quad \text{Min} \sum_{i,j \in N} c_{ij}\, x_{ij} \qquad \text{(II.1)}$$

sujeito a

$$\sum_{i \in N,\ i \neq j} x_{ij} = 1 \quad (j \in N) \qquad \text{(II.2)}$$

$$\sum_{j \in N,\ i \neq j} x_{ij} = 1 \quad (i \in N) \qquad \text{(II.3)}$$

$$\sum_{i \in S} \sum_{j \in \overline{S}} x_{ij} \geq 1 \quad (2 \leq |S| \leq n-2,\ S \subset N) \qquad \text{(II.4)}$$

$$x_{ij} = \{ \emptyset, 1 \} \qquad (i \neq j \text{ e } i, j \in N) \qquad (II.5)$$

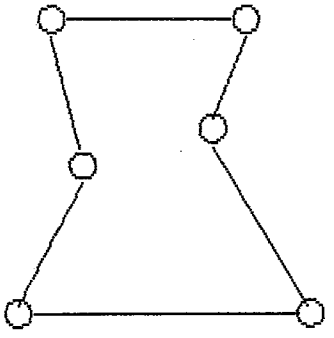
Nesta formulação, as restrições (II.2), (II.3) e (II.5) descrevem o Problema de Alocação (ou Designação). As restrições (II.4) eliminam as ocorrências de subrotas, obrigando a existência de pelo menos um arco interligando uma partição $(S,\bar{S})$ de N, i.é, evitam obter um grafo desconexo como solução.

LITTLE et al. (1963) apresentam uma estratégia de "bounding" que relaxa as restrições (II.4). O Problema de Alocação resultante é resolvido evitando-se a formação de subrotas impondo penalidades sobre os arcos que as formariam.

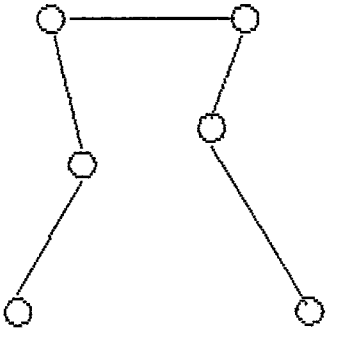
CHRISTOFIDES & EILON (1969) sugerem como limitante inferior uma **Árvore Geradora de Peso Mínimo**, descrevendo a sua justificativa da seguinte maneira:

> " A rota do caixeiro viajante através de n pontos é um ***Ciclo Hamiltoniano***, isto é, um ciclo onde cada nó possui grau dois. Uma **Árvore Geradora** é uma configuração de n-1 arcos passando por n pontos e uma **Árvore Geradora de Peso Mínimo**, ou simplesmente Árvore Geradora Mínima, é uma Árvore Geradora onde a soma de seus pesos é mínima. A figura II.4(a) mostra um exemplo de uma rota mínima de caixeiro viajante. A árvore geradora mostrada em (b) é obtida pela

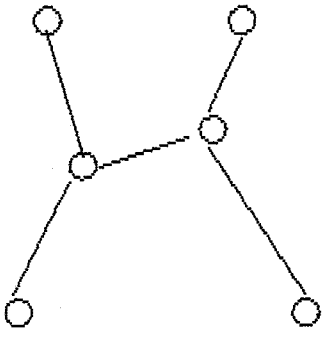
remoção de um arco de (a), portanto (b) é um Limitante Inferior ("Lower Bound") para (a). A árvore de peso mínimo em (c) também é um Limitante Inferior para (a). Como a configuração em (c) consiste de n-1 ligações, um Limitante melhor para a rota do caixeiro viajante pode ser obtido pela adição de um arco conveniente, por exemplo, o menor arco do grafo, como está mostrado em (d). Alternativamente um ponto fictício pode ser adicionado, colocando-o na mesma posição de um já existente, porém, designando-se uma distância infinita entre eles (e). Desta forma converte-se o conjunto de n pontos em um de n+1 pontos. A árvore de peso mínimo dos n+1 pontos é um "Lower Bound" para a rota mínima do caixeiro viajante dos n pontos originais."



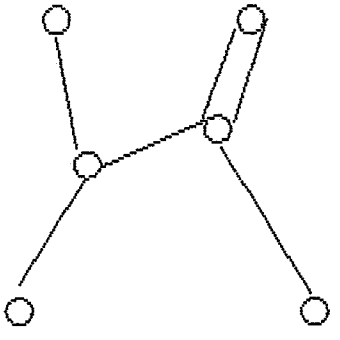
(a)



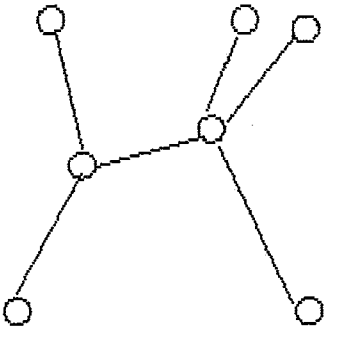
(b)



(c)



(d)



(e)

figura II.4 - A Árvore Geradora Mínima como um "bound" para o PCV.

II.2.2 - Branching nas Rotas

Em CHRISTOFIDES (1976) é descrito um algoritmo B-B onde as rotas viáveis são geradas como e quando desejadas. Portanto cada nó da árvore de busca corresponde à definição de uma nova rota de veículos.

O estado da busca em um estágio h de uma árvore de B-B pode ser representado por uma lista ordenada H, da seguinte maneira:

$$H = \{S_{j_1}(x_{i_1}),\ S_{j_2}(x_{i_2}),\ \ldots,\ S_{j_h}(x_{i_h})\}$$

onde $S_{j_r}(x_{i_r})$ é uma rota viável que inclui um cliente específico x_{i_r} e outros clientes que ainda não foram incluídos nas rotas anteriores $S_{j_1}(x_{i_1}),\ldots,\ S_{j_{r-1}}(x_{i_{r-1}})$.

Seja $F(h)$ o conjunto dos clientes livres, i.é., que não pertencem a nenhuma rota até o estágio h, e seja $\bar{F}(h) = N' - F(h)$ o conjunto dos clientes que já pertencem a uma rota em h.

Um "branching" de algum nó h envolve a escolha de um cliente $x_{i_{h+1}} \in F(h)$ e a construção de uma lista $P(x_{i_{h+1}})$ de todas as rotas viáveis passando por este cliente. Parece evidente que quanto menor for o número de ramificações ("branchs") possíveis em um nó h mais eficiente será a busca. Portanto, a escolha de $x_{i_{h+1}}$ deve

ser feita de forma a tornar a lista $P(x_{i_{h+1}})$ a menor possível. Este parece ser o caso em que $x_{i_{h+1}}$ é escolhido como o cliente mais afastado do depósito ou o que possui uma demanda muito alta.

Uma vez alcançado o fundo da árvore de busca, isto ocorre em um estágio m quando $F(m) = \phi$ (onde m é o número de rotas da solução ótima), a lista H contém uma solução para o problema que consiste de m rotas. Nesta altura um "***backtracking***" deve ser realizado para considerar as possibilidades ainda não analisadas.

Entretanto, nem todos os nós produzidos por $P(x_{i_{h+1}})$ necessitam ser explorados, em geral é possível mostrar que um nó específico pode ser abandonado. Isto pode ser feito em uma das seguintes circunstâncias:

(i) A rota que contém $x_{i_{h+1}}$ viola uma das restrições do problema;

(ii) Os veículos ainda não utilizados não podem suprir as demandas dos clientes de F(h+1), i.é, a capacidade total destes veículos é menor que o volume das demandas dos clientes livres;

(iii) Sejam Z^* o custo da melhor solução viável conhecida, C(S) o custo ótimo para suprir os clientes de $S \subseteq N'$, $\bar{C}(S)$ um "lower bound" para

$C(S)$ e $C^{+}(S)$ um "upper bound". Então o nó $h+1$ da árvore de busca não precisa ser criado, a partir do nó h, se:

$$C(\bar{F}(H)) + C(\bar{F}(h+1) - \bar{F}(H)) + C^{-}(F(h+1)) \geq Z^{*} \ ;$$

(iv) Se $(h+1)$ é dominado por outro descendente imediato $(h'+1)$ de h, i.é.:

$$C(\bar{F}(h'+1)-\bar{F}(H))+C^{+}(F(h'+1)) \leq C(\bar{F}(h+1)-\bar{F}(H)+C^{-}(F(h+1))).$$

Nos ítens acima é assumido que limitantes superiores e inferiores para os subproblemas restantes, definidos pelos clientes livres, podem ser calculados em qualquer estágio. Limitantes inferiores podem, como já vimos, ser calculados pela relaxação de alguma das restrições e os limitantes superiores podem ser obtidos por alguma heurística.

Em CHRISTOFIDES et al. (1981) os autores apresentam duas maneiras de calcular "lower bounds" para o PPV-básico, que mostramos em seguida

II.2.3 - Bound a Partir da k-DCT

Uma **"k-Degree Centre Tree"** (k-DCT) é uma árvore onde o nó raiz possui grau **k**. Este "bound"

basea-se no fato de que o valor da solução ótima do Problema do Caixeiro Viajante Múltiplo (**m-PCV**) é limitado pelo peso total de uma **k-DCT**.

Podemos observar que o conjunto dos arcos pertencentes à solução ótima pode ser particionado em três subconjuntos, a saber:

(i) A_1: arcos que formam uma k-DCT, onde $k = 2m - y$ $y \leq m$;

(ii) A_2: y arcos incidentes ao nó 1 (depósito);

(iii) A_3: m - y arcos não incidentes ao nó 1.

Para derivarmos o "bound" a partir da **k-DCT** é necessário antes apresentarmos uma nova formulação para o problema.

Sejam:

.$\xi_I^t = \begin{cases} 1 & \text{se } I \in A_t \\ \emptyset & \text{c.c.} \end{cases}$, onde t= 1,2,3 e $I \in A$;

.y: número de arcos incidentes ao depósito;

.c_I: o custo do arco I;

.A^i: o conjunto dos arcos incidentes ao nó i;

.$(S,\bar{S})$: o conjunto de todos os arcos com um vértice em S e o outro em $\bar{S}$.

Então, podemos formular o problema da seguinte maneira:

$$(\text{m-PCV})\ \text{Min} \sum_{I \in A} c_I(\xi_I^1 + \xi_I^2 + \xi_I^3) \qquad (II.6)$$

sujeito a:

$$\sum_{I \in (S,\bar{S})} \xi_I^1 \geq 1 \qquad (S \subseteq A,\ S \notin \phi) \qquad (II.7)$$

$$\sum_{I \in A_1} \xi_I^1 = 2m - y \qquad (II.8)$$

$$\sum_{I \in A} \xi_I^1 = n - 1 \qquad (II.9)$$

$$\sum_{I \in A_1} \xi_I^2 = y \qquad (II.10)$$

$$\sum_{I \in A-A^1} \xi_I^3 = m - y \qquad (II.11)$$

$$\sum_{I \in A^i} (\xi_I^1 + \xi_I^2 + \xi_I^3) = 2 \qquad (i \in N') \qquad (II.12)$$

$$\xi_I^t \in \{0,1\} \ ; \ t=1,2,3 \ \text{e} \ I \in A \qquad (II.13)$$

$$y \geq 0 \ \text{e inteiro} \qquad (II.14)$$

A restrição (II.7) impõe a conectividade da **k-DCT**, (II.8) define o grau do vértice 1 (depósito), (II.9),(II.10) e (II.11) especificam o número de arcos necessários e (II.12) garantem que os nós de N' tenham grau igual a 2.

Finalmente, (II.13) e (II.14) impõe a integralidade às variáveis de decisão.

Na formulação acima o número de veículos foi considerado como uma variável de decisão **y**, porém se fixarmos o seu valor em m, estaremos considerando uma frota cujo tamanho é fixado a priori. Notamos também que a formulação acima não considera as restrições de capacidade e distância, sendo portanto o Problema do Caixeiro Viajante Múltiplo (**m-PCV**).

O "bound" é obtido através da Relaxação Lagrangeana das restrições (II.12), decompondo o problema em três subproblemas P_1, P_2 e P_3. A função objetivo do Problema Lagrangeano fica:

$$V^t(\lambda,y)= \Sigma_{I\in A}\ \bar{c}_I\omega_I - 2\ \Sigma_{i\in N'}\ \lambda_i$$

onde:

.$\lambda_i \geq \emptyset$, $i\in N'$ é o vetor de penalidades associadas com as restrições relaxadas, e $\lambda_1 = \emptyset$;

.$\bar{c}_I = c_I + \lambda_{i_I} + \lambda_{j_I}$ onde i_I e j_I são os dois nós adjacentes ao arco I;

.t = 1,2 ou 3;

.$\omega_I = \{\xi_I^t;\ t = 1,2\ 3\}$ de acordo com cada problema.

Para um dado valor de y e λ, sejam:

.$V^1(\lambda,y)$ o valor da solução ótima de P_1 dado pelas restrições (II.7), (II.8), (II.9) e (II.13);

.$V^2(\lambda,y)$ o valor da solução ótima de P_2 dado pelas restrições (II.10) e (II.13);

.$V^3(\lambda,y)$ o valor da solução ótima de P_3 dado pelas restições (II.11) e (II.13).

Então um "lower bound" para o (m-PCV) é

$$\beta = \text{Máx}_{m_1 \leq y \leq m} \{\text{Máx}_{\lambda \geq 0} \Sigma_{t=1,3} V^t(\lambda,y)\} \qquad (II.15)$$

onde m_1 representa um limitante sobre o número de rotas na solução ótima.

O "bound" acima naturalmente também vale para o PPV-básico. Mais que isto, podemos melhorá-lo quando consideramos as restrições de capacidade e distância (ou tempo) percorrida por cada veículo. Esta melhoria baseia-se na escolha de um melhor valor de m_1 para o PPV. Isto é feito da seguinte maneira (LAPORTE & NOBERT, 1985):

suponha os clientes ordenados em ordem decrescente de suas demandas d_i. Como m_1 rotas podem no máximo suprir uma quantidade $\Sigma_{i=1,m_1} d_i$ e os veículos restantes devem estar aptos a suprir as

demandas ainda não atendidas, temos que:

$$(m - m_1)\ D \geq \sum_{i=m_1+1,n} d_i \qquad (II.16)$$

Analogamente, cada cliente i contribui com uma quantidade de pelo menos $u_i = \delta_i + 1/2(c_{ii_1} + c_{ii_2})$ para o comprimento de uma rota, onde δ_i **é o tempo gasto no atendimento do cliente** i, i_1 e i_2 são as duas cidades mais próximas de i. Então, se as cidades forem ordenas em ordem decrescente de u_i, m_1 deve satisfazer a:

$$(m - m_1)\ L \geq \sum_{i=m_1+1,n} u_i \qquad (II.17)$$

Portanto, podemos tomar m_1 como sendo o maior valor que satisfaz (II.16) e (II.17).

A seguir descrevemos os procedimentos sugerido em CHRISTOFIDES et al. (1981) para o cálculo de cada $V^t(\lambda,y)$, $t = 1,2$ e 3, usado no cálculo do "bound" em (II.15).

- <u>Cálculo de $V^1(\lambda,y)$</u>

Para um dado valor de λ e de y, resolver P_1 significa determinar uma k-DCT de peso mínimo. Seja T_k^* a solução de P_1. T_k^* pode ser obtida através do cálculo da Árvore Geradora Mínima do grafo, verificando-se o grau g do vértice 1 (o depósito) da seguinte maneira:

(i) se $g = k$ então o problema está resolvido;

(ii) se $g < k$ então uma penalidade $\mu > 0$ é dada ao vértice i, e os custos c_{ij} são trocados para $c_{ij} - \mu$;

(iii) se $g > k$ então uma penalidade $\mu < 0$ é dada ao nó i e os custos c_{ij} tornam-se $c_{ij} - \mu$.

Nos casos (ii) e (iii) a Árvore Geradora Mínima é recalculada, e o procedimento é repetido até que $g = k$.

- <u>Cálculo de $V^2(\lambda, y)$</u>

A solução de P_2 é dada pelos y arcos incidentes ao depósito, isto é que pertencem a A_2, de menor comprimento $\bar{c}_{ij}$. Uma boa escolha destes y arcos pode ser obtida pelo seguinte procedimento:

.defina $\bar{c}_1^r$ e $\bar{c}_2^r$ como sendo os custos dos dois arcos que ligam uma rota r ao depósito; suponha que $\bar{c}_2^r \geq \bar{c}_1^r$ e note que quando $\bar{c}_1^r = \bar{c}_2^r$ a rota é formada por um único cliente;

.Ordene crescente e lexicograficamente as rotas pelo vetor $(\bar{c}_2^r, \bar{c}_1^r)$, renumerando-as de maneira que a

rota r se refira à r-ésima rota da ordenação. Como para cada uma das y rotas podemos escolher para elemento de A_2 o mais longo entre os dois arcos que ligam cada rota ao depósito, temos que o custo de A_2 será sempre menor ou igual a $\sum_{r=1,y} \bar{c}_2^r$;

.Construa uma outra lista dos arcos adjacentes ao depósito, ordenada crescentemente pelos custos $\bar{c}_{1j}$, repetindo os primeiros m_1 custos da lista de forma que estes apareçam duas vezes na lista. Defina $h(p)$ como o custo na p-ésima posição desta lista. Então, temos que para qualquer solução do **m-PCV**, acima formulado, $\bar{c}_2^1 \geq h(2)$, $\bar{c}_2^2 \geq h(4)$, $\bar{c}_2^3 \geq h(6)$, etc. e em geral $\bar{c}_2^r \geq h(2r)$ para qualquer $r = 1,\ldots,m$;

Então, temos que:

$$L^2(\lambda,y) = \sum_{r=1,y} h(2r) \leq \sum_{r=1,y} \bar{c}_2^r$$

é um limitante inferior para o custo de A_2 e pode ser usado em (II.15) no lugar de $V^2(\lambda,y)$.

- <u>Cálculo de $V^3(\lambda,y)$</u>

A solução de P_3 é dada pelo somatório dos custos $\bar{c}_I$ dos $(m-y)$ arcos de menor custo não incidentes ao depósito (i.é., arcos que pertencem a A_3).

Seja $\bar{c}_i[1]$ o custo do arco incidente ao nó i de menor custo. Não considerando o depósito, suponha que as quantidades $c_i[1]$ estejam ordenadas na ordem crescente e que os clientes tenham sido renumerados de forma que i se refira ao i-ésimo desta lista. Desta maneira, $V^3(\lambda,y)$ é calculado por:

$$V^3(\lambda,y) = \sum_{i=1,m-y} \bar{c}_i[1] \qquad (II.18)$$

- O Algoritmo para o cálculo do bound pela k-DCT

Para definir por completo a estratégia de "bound", baseada na relaxação da k-DCT, é apresentado em CHRISTOFIDES et al. (1981) um algoritmo onde:

.y é inicializado com o valor m_1 (dado por (II.16) e (II.17)) e sempre que necessário é incrementado de uma unidade;

.os valores λ_i, i=1,n , são calculados pela fórmula dada pelo **método do subgradiente** até um número de iterações fixado à priori.

A seguir apresentamos o algoritmo.

PØ. (inicialização)

defina Z_L^* como o melhor lower bound e Z_U^* o valor da melhor solução encontrada até o momento. Faça $Z_L^* = \emptyset$ e $y = m_1$;

P1. (Início passo iterativo)

Faça kount= 1, $\lambda_i = 0$ para i=1,...,n , k= 2m-y;

P2. (k-DCT)

Calcule T_k^* da maneira que mencionamos anteriormente;

P3.(Arcos adicionais)

Calcule $L^2(\lambda,y)$ e $V^3(\lambda,y)$;

P4. (Grafo PPV)

Forme o grafo G pela união dos arcos encontrados nos passos P2 e P3 e faça Z_L= custo total deste grafo, que é um "lower bound" para a solução do **PPV**.

Se $Z_L^* < Z_L$ faça:

.$Z_L^* = Z_L$;

.Se $Z_L^* \geq Z_U^*$ PARE (este nó deve ser abandonado, realize um backtracking no algoritmo principal)

.Senão (se $Z_L^* < Z_U^*$) vá para o passo P5;

Se $Z_L^* \geq Z_L$ e kount= No. máximo de iterações permitidas vá para o passo P7;

Senão faça kount = kount + 1 e vá para o passo P5.

P5. (Penalidades λ_i)

Se o grau g_i do vértice x_i do grafo G é igual a dois e $g_1 = 2m$, PARE (Z_L^* é o melhor "lower bound" que pode ser obtido por este procedimento);

caso contrário calcule as penalidades λ_i da seguinte maneira:

$$\lambda_i = \lambda_i + \alpha \frac{(Z_U^* - Z_L)}{[\ \Sigma_{j \in N'} (d_j - \sigma_j)^2\ ]^{1/2}} (d_i - \sigma_i)$$

onde $\sigma_i = 2$ se $i \neq 1$ e $\sigma_i = 2m$ se $i = 1$ e α é uma cte..

P6. (Atualize matriz de custos)

Faça $c_{ij} = c_{ij} + \lambda_i + \lambda_j \quad \forall\ i,j \in N$ e vá para o passo P2;

P7. (Atualize y)

Se y= m PARE, senão faça y= y + 1 e vá para o passo P1.

II.2.4 - Bound a Partir da q-Rota Mínima

O segundo "bound" desenvolvido em CHRISTOFIDES et al. (1981) é baseado em uma rota especial, denominada q-rota, que definimos abaixo.

Seja W o conjunto de todas as cargas (demandas) que podem existir em qualquer rota de veículo, ou seja:

$$W = \{q \;/\; \sum_{i \in N'} d_i \xi_i = q \leq D, \text{ para algum } \xi_i \in \{0,1\} \}$$

(II.19)

Considere que os elementos de W estão ordenados crescentemente e que $q(I)$ é o valor do I-ésimo elemento de W. Uma q-rota de menor custo, $\Psi_I(i)$, é uma rota que:

(i) passa pelo nó i;

(ii) inicia e termina no depósito;

(iii) não possui ciclos do tipo (i_1, i_2, i_1) ;

(iv) possui demanda total $q(I)$.

Para derivar um "bound" a partir desta **q-rota**, é necessário que formulemos o **PPV** da seguinte maneira:

$$\textbf{(PPV1)} \quad \textbf{Min} \sum_{r=1,r'} c_r y_r \qquad \text{(II.20)}$$

sujeito a:

$$\sum_{r \in R_i} y_r = 1 \qquad (i=1,n) \qquad \text{(II.21)}$$

$$\sum_{r=1,r'} y_r = m \qquad \text{(II.22)}$$

$$y_r \in \{0,1\} \qquad \text{(II.23)}$$

onde:

.r = 1,r' é o índice das r' rotas viáveis;

.c_r é o custo da rota r;

.R_i é o conjunto de índices de rotas que visitam o cliente i;

.$y_r = \begin{cases} 1 & \text{se a rota r pertence à solução;} \\ \emptyset & \text{c.c.} \end{cases}$

Podemos reformular o problema acima substituindo y_r por:

$$y_r = \frac{1}{K_r} \sum_{i \in M_r} \xi_{ir} d_i \quad ; \quad \xi_{ir} \in \{\emptyset, 1\} \ ,$$

onde M_r é o conjunto de índices dos clientes da rota r e $K_r = \sum_{i \in M_r} d_i$ é a carga total da rota r, e substituindo a restrição (II.21) pelas seguintes :

$$\sum_{r \in R_i} \xi_{ir} = 1 \qquad (i = 1,n) \qquad \text{(II.24)}$$

$$\xi_{ir} = \frac{1}{K_r} \sum_{j \in M_r} \xi_{jr} \, d_j \qquad (i \in M_r \ ; \ r = 1,r') \qquad \text{(II.25)}$$

A restrição (II.25) garante que $\xi_{ir} = 1$ se e só se $\xi_{jr} = 1 \ \forall \ j \in M_r$ e portanto $y_r = 1$.

Vamos relaxar este novo problema removendo as restrições (II.25) e trocando o conjunto M_r pelo conjunto completo de clientes N', para obter a seguinte formulação:

$$\text{(PPV2)}\quad \text{Min} \sum_{i\in N} \sum_{I=1,w} \bar{c}_{iI}\, \xi_{iI} \qquad \text{(II.26)}$$

sujeito a:

$$\sum_{I=1,w} \xi_{iI} = 1 \quad (i = 1,\ldots,n) \qquad \text{(II.27)}$$

$$\sum_{i\in N'} \sum_{I=1,w} \frac{d_i}{q(I)}\, \xi_{iI} = m \qquad \text{(II.28)}$$

$$\sum_{i\in N'} \sum_{I=1,w} d_i \xi_{iI} = D_T \qquad \text{(II.29)}$$

$$\xi_{iI} \in \{0,1\} \qquad \text{(II.30)}$$

onde $\bar{c}_{iI} = c_{iI}\, d_i/q(I)$, c_{iI} sendo o custo da rota que passa por i com carga $q(I)$; $D_T = \sum_{i\in N'} d_i$ e $w = |W|$.

É evidente que o custo $\psi_I(i)$, custo da rota mínima que passa por i com carga $q(I)$, é um "lower bound" para c_{iI}. Portanto a solução do problema

$$\text{Min} \sum_{i\in N'} \sum_{I\in W} b_{iI}\xi_{iI} \qquad \text{(II.31)}$$

sujeito a (II.27), (II.28), (II.29) e (II.30), onde $b_{iI} = \psi_I(i)\, d_i/q(I)$ é um lower bound para o PPV.

Obs.: Note que b_{iI} é um lower bound para $\bar{c}_{iI}$ obtido pela relaxação das restrições que garantem que em uma solução viável o grau de cada nó, diferente do depósito, é

igual a dois.

Finalmente, podemos facilmente calcular um "lower bound" para o **PPV** se ignorarmos as restrições (II.28) e (II.29) do problema acima definido, resultando em:

$$\beta = \sum_{i \in N'} \; \min_{l \in W} \; [b_{il}].$$

Uma alternativa mais refinada seria calcular o "lower bound" através de uma Relaxação Lagrangeana das restrições (II.28) e (II.29).

II.3 - Algoritmos de Programação Dinâmica

Os principais elementos de um algoritmo de Programação Dinâmica (**P.D.**) são **Estados**, **Transições** entre estados e **Equações de Recorrência** para determinar o valor da Função Objetivo em cada estado. A formulação de um problema de percursos de veículos através da Programação Dinâmica apresenta uma dificuldade comum a vários problemas de otimização combinatória: o número extremamente elevado de estados torna sua resolução inviável na prática.

Afim de contornar esta dificuldade CHRISTOFIDES et al. (1981b) introduziram um método, denominado "**Relaxação do Espaço de Estados**", para reduzir o número de estados do problema.

II.3.1 - Relaxação do Espaço de Estados

Relaxar o espaço de estados associado a uma dada recursão de programação dinâmica consiste em reduzir o número de estados de tal forma que a solução da recursão relaxada determine um limitante ("bound") para o valor ótimo do problema; este "bound" pode ser então utilizado em um procedimento B-B para solucionar o problema original.

A redução do número de estados é feita através de uma função $g(.)$ que mapeia valores de um espaço de estados S em um espaço de estados G de menor cardinalidade. Portanto, se existir uma transição de um estado S_1 para um estado S_2 no espaço de estados original, então deve existir uma transição de $g(S_1)$ para $g(S_2)$ no espaço de estados relaxado G.

A relaxação do espaço de estados pode ser bem entendida através do clássico **Problema do Caixeiro Viajante (PCV)**. Vamos considerar a seguinte formulação do **PCV** em termos de Programação Dinâmica **(P.D.)**.

Seja $f(S,i)$ o custo do caminho mais curto que inicia no depósito 1, visita cada nó de S e termina no nó $i \in S$. A recursão da **P.D.** é dada por:

$$f(S,i) = \min_{j \in S-\{i\}} \{f(S-\{i\},j) + c_{ji}\} \qquad (II.30)$$

para todo $S \subseteq N'$ e $\forall\, i \in S$.

A inicialização da recorrência é dada por

$$f(\{i\},i) = c_{1i} \quad \forall\, i.$$

Então, a solução ótima do problema é

$$\min_{i \in N'} \{f(N',i) + c_{i1}\}.$$

Seja $g(.)$ a função que mapeia o domínio de (S,x) para algum outro espaço $(g(S),x)$ de menor cardinalidade. Então, a relaxação da recursão (II.30)para o PCV torna-se:

$$f(g(S),x) = \min_{y \in S'} \{f(g(S-\{x\}),y) + c_{yx}\} \qquad (II.31)$$

onde $\Gamma^{-1}(x) \supseteq S' \supseteq S-\{x\}$. A inicialização é dada por:

$$f(w,y) = \begin{cases} c_{1y} & \text{se } w = g(\{y\}), \\ \infty & \text{caso contrário.} \end{cases}$$

Listamos abaixo cinco funções $g(.)$ examinadas em CHRISTOFIDES et al. (1981b), que podem ser utilizadas na relaxação descrita acima.

(i) $g(S) = |S|$;

(ii) $g(S) = \sum_{i \in S} d_i$;

(iii) $g(S) = (|S|, \sum_{i \in S} d_i)$;

(iv) $g(S) = (\sum_{i \in S} d_i^1 , \sum_{i \in S} d_i^2)$, onde d_i^1 e d_i^2 são inteiros tais que $d_i^1 + d_i^2 \geq 1$;

(v) $g(S) = (\sum_{i \in S_1 \cap S} d_i , \sum_{i \in S_2 \cap S} d_i , \ldots , \sum_{i \in S_r \cap S} d_i)$

onde $S_1, \ldots, S_r$ representam uma partição do conjunto N'.

II.3.2 - Formulações do PPV em Programação Dinâmica

É importante observar que a eficácia da relaxação do espaço de estados em produzir limitantes é relativa a uma dada formulação de P.D.. Nesta seção apresentamos três formulações para o PPV com restrições de capacidade. Na seção seguinte derivamos os limitantes a partir da relaxação do espaço de estados para cada uma destas formulações.

a) Formulação 1

Sejam $f(m',S)$ o menor custo para suprir um conjunto S de clientes usando apenas m' veículos; $v(S)$ a solução para o PCV definido pelos clientes pertencentes a S e o depósito 1. Então a fórmula de recursão da P.D. fica:

$$f(m',S) = \underset{U \subset S}{\text{Min}} \{f(m'-1,S-U) + v(U)\} \qquad (II.32)$$

sujeito a:

$$\sum_{i \in S} d_i - (m'-1)D \leq \sum_{i \in U} d_i \leq D \quad p/\ m' = 2,m \qquad (II.33)$$

onde $S \subseteq N'$ deve satisfazer a:

$$D_T - (m-m')D \leq \sum_{i \in S} d_i \leq m'D \qquad (II.34)$$

onde $D_T = \sum_{i \in N'} d_i$.

A recursão é inicializada por $f(1,S) = v(S)$.

b) Formulação 2

Vamos ver uma relaxação alternativa que contém alguma redundância de estados em relação a anterior. Esta redundância é útil quando realizamos a relaxação do espaço de estados (uma analogia à relaxação lagrangeana pode ser

feita para uma primeira justificativa deste fato).

Seja $f(m',S,k)$ o menor custo para suprir os clientes de S, usando m' veículos, com os últimos clientes das correspondentes m' rotas estando entre os clientes $2,\ldots,k$. E seja $v(S,k)$ a solução do PCV no cojunto S onde o último cliente da rota é k. Neste caso a recursão é dada por:

$$f(m',S,k) = \mathbf{Min}\{f(m',S,k-1),\ \min_{U \subset S} \{f(m'-1,S-U,k-1)+v(U,k)\}\} \qquad (II.35)$$

sujeito a (II.33).

c) Formulação 3

Seja $f(m',S,\bar{d},i)$ o menor custo para suprir os clientes de S usando m' veículos onde as primeiras $(m'-1)$ rotas estão fechadas e a última rota está aberta, tem uma demanda de $\bar{d}$ e termina no cliente i. A recursão desta P.D. é:

$$f(m',S,\bar{d},i) = \min_{j \in S \cap \Gamma^{-1}(i)} \{f(m',S-\{i\},\bar{d}-d_i,i) + c_{ji}\} \qquad (II.36)$$

para $d_i \leq \bar{d} \leq D$ e $m'=1,\ldots,m$.

As funções são inicializadas por :

$$f(1,\{i\},\bar{d},i) = \begin{cases} c_{1i} & \text{para } \bar{d} = d_i \\ \infty & \text{c.c.} \end{cases} \qquad (II.37)$$

O valor ótimo do **PPV** é então dado por:

$$\min_{i\in\Gamma^{-1}(1),\bar{d}} \{f(m,N',\bar{d},i) + c_{i1}\} \qquad (II.38)$$

onde $D_T - (m-1)D \leq \bar{d} \leq D$.

II.3.3 - Relaxação do Espaço de Estados e Bounds para o PPV

As mesmas funções de mapeamento utilizadas no PCV podem ser usadas para relaxar as retrições dadas acima, no caso do **PPV**. As variaveis S, do vetor de estado em todas as formulações acima, podem ser relaxadas através das funções **g(S)** dadas na seção (II.2.1).

a) Relaxação da formulação 1

Considere $g(S) = q = \sum_{i\in S} d_i$, então a relaxação da recursão (II.32) fica:

$$f(m',q) = \min_{p} \{f(m'-1,q-p) + \bar{v}(p)\} \qquad (II.39)$$

onde $D_T-(m-1)D \leq p \leq \min\{q,D\}$ e $\bar{v}(p)$ é o custo mínimo do circuito C para o qual $\sum_{i\in C} d_i = p$. Como o cáloulo de $\bar{v}(p)$ é um problema difícil , define-se uma função $\bar{v}(p)$

para ser um Limitante Inferior para este custo:

$$\bar{v}(p) = \min_{i \in \Gamma^{-1}(1)} \{f(p,i) + c_{i1}\} \qquad (II.40)$$

onde:

$$f(p,i) = \min_{j \in E^{-1}(p,i)} \{f(p-p_i, j) + c_{ji}\} \qquad (II.41)$$

onde:

$$f(p,i) = \begin{cases} c_{1i} & \text{se } q=d_i \\ \infty & \text{caso contrário} \end{cases} ;$$

$$E^{-1}(p,i) = \{j \;/\; i \in \Gamma(j),\ p-d_i \geq d_j\}.$$

Um limitante inferior para o PPV é então dado por $f(m,D_T)$.

b) Relaxação da formulação 2

Um bound melhor pode ser obtido a partir da recursão (II.35). Usando-se a mesma função $g(S) = q$ chega-se à seguinte relaxação de (II.35):

$$f(m',q,k) = \min \{f(m',q,k-1),\ \min_{p} \{f(m'-1,q-p,k-1)+\bar{v}(p,k)\}\} \qquad (II.42)$$

para $D_T-(m-1)D \leq p \leq \min \{q,D\}$

onde $\bar{v}(p,k) = f(p,k) + c_{k1}$.

O limitante inferior será dado por

$f(m,D,n) \geq f(m,D_T)$.

c) Relaxação da formulação 3

Utilizando-se mais uma vez da função de mapeamento $g(s)= q$ para relaxar a recursão (II.38), obtemos:

$$f(m',q,\bar{d},i) = \min_{j\in\Gamma^{-1}(i)} \{f(m',q-d_i,\bar{d}-d_i,j) + c_{ji}\} \quad (II.43)$$

O "lower bound" obtido diretamente da recursão acima é:

$$\min_{i\in\Gamma^{-1}(i)} \{f(m,D_T,\bar{d},i) + c_{i1}\}$$

onde $D_T-(m-1)D \leq \bar{d} \leq D$.

II.4 - Algoritmos Baseados em Particionamento de Conjuntos

Podemos formular o **PPV** como uma generalização do Problema de Particionamento de Conjunto (Set Particioning), como sugerem BALINSKI & QUANDT (1964).

Consideremos $R = \{r \ / \ r = 1,\ldots,m\}$ como sendo o conjunto das rotas viáveis de um **PPV**. Sejam a_{ir} um coeficiente que toma o valor 1 se, e só se, a cidade i pertence à rota r, c_r^* o custo ótimo da rota r, e a variável $x_r = 1$ se a rota r pertence à

solução ótima ou $x_r = 0$ caso contrário. Podemos então escrever o **PPV** da seguinte maneira:

$$\textbf{(PPV3) Min} \quad \sum_{r \in R} c_r^* x_r \qquad \text{(II.44)}$$

sujeito a:

$$\sum_{r \in R} a_{ir} x_r = 1 \quad (i \in N') \qquad \text{(II.45)}$$

$$x_r \in \{0,1\} \qquad \text{(II.46)}$$

Existem duas dificuldades associadas a esta formulação:

(i) o excessivo número de variáveis binárias que um problema real pode alcançar;

(ii) a dificuldade para calcular todos os valores c_r^*. Por exemplo, no **PPV-básico**, cada rota r corresponde a um conjunto de cidades S_r que satisfaz a:

(a) $\sum_{i \in S_r} d_i \leq D$ (restrição de capacidade);

(b) $\sum_{i,r \in S_r} c_{ir} \leq L$ (restrição de distância).

Portanto para obter o valor de cada c_r^* é preciso resolver um Problema do Caixeiro Viajante sobre S_r.

Podemos enfrentar estas dificuldades através da

técnica de **Geração de Colunas**. No método Simplex padrão, a cada iteração devemos escolher uma variável não-básica para entrar na base, logo, para resolvermos o **PPV3** acima precisamos ter em mãos as colunas $a_{.r}$ relativas a todas as rotas viáveis possíveis, que como já vimos exige um esforço computacional muito elevado. Na técnica de geração de colunas nós só calculamos as colunas na medida em que precisamos, ou seja, uma nova coluna - relativa à variável entrante na base - de custo marginal mínimo é gerada pela solução de um subproblema apropriado em cada iteração do Simplex. Se o custo marginal é negativo então a coluna é adicionada ao problema de Programação Linear, o problema é reotimizado e a geração de coluna é novamente aplicada; se, ao contrário, o custo marginal é positivo siginifica que a solução corrente é a ótima.

Esta técnica é utilizada por vários autores para tratar de **PPV's** que podem ser formulados por Particionamento de Conjuntos. Vejamos alguns destes exemplos.

FOSTER & RYAN (1976) sugerem que um esquema de geração de colunas onde cada rota é obtida por Programação Dinâmica pode teoricamente ser desenvolvido. Porém este esquema possui uma taxa de convergência muito lenta, além de raramente fornecer uma solução inteira.

Tentando evitar estes problemas, FOSTER & RYAN (1976) apresentam uma alternativa de solução que se

baseia no seguinte procedimento:

(i) Definir uma nova região viável, mais restrita que a original, de maneira a diminuir o número de rotas a serem exploradas, consequentemente diminuindo o número de colunas a serem geradas;

(ii) Usando informações providas deste modelo, que passamos a chamar de "Problema Restrito", relaxar progressivamente as restrições adicionais sobre as novas rotas viáveis de maneira a encontrar uma solução ótima para o problema original.

Podemos restringir a região viável observando que:

(a) as rotas servem um setor da região com centro no depósito. Pontos de entrega dentro de um setor são raramente desviados; exceções resultam de restrições na estrutura das rotas (p.ex., capacidade e distância);

(b) rotas adjacentes raramente se cruzam;

(c) para algum subconjunto de pontos de entrega, a rota ótima é a solução do Problema do Caixeiro Viajante para aquele subconjunto de pontos de entrega.

Portanto, se restringirmos a composição de R a rotas com estes atributos, nós efetivamente temos que

resolver um modelo mais "restrito" do **PPV**. O problema de definir conjuntos restritos de rotas se reduz a designar pontos de entregas às rotas, baseadas em (i) e (ii) e então achar a solução do PCV para cada rota.

Existem várias maneiras de se realizar a designação acima. FOSTER & RYAN (1976) desenvolveram um algoritmo que segue o seguinte procedimento.

Os pontos de entrega são ordenados radialmente a partir do depósito e todos os subconjuntos contíguos desta ordenação são considerados candidatos para uma rota. Subconjuntos cuja capacidade total excedem o limite para uma rota são rejeitados e os remanescentes são submetidos a uma heurística que solucione o PCV. A rota resultante é então adotada como uma rota viável, contanto que não exceda o limite **L** adotado para a distância máxima permitida. A solução viável produzida desta maneira fornece rotas do tipo radial (semelhante a pétalas de uma flôr).

O procedimento acima visa obter boas soluções para o **PPV**-básico dentro de um tempo computacional razoável. Como resultado obtemos uma solução viável restrita a um conjunto de rotas promissoras. A mesma filosofia pode ser adotada no processo de relaxamento da região viável do Problema Restrito. Ou seja, o processo é direcionado de maneira que a busca seja realizada em áreas onde exista uma provável melhoria. Uma relaxação que obteve sucesso entre as investigadas pelos autores é a que troca um cliente entre

duas rotas de maneira a reduzir a distância percorrida.

Uma característica que merece destaque é a propriedade de "quase-integralidade" da estrutura da região viável do problema Restrito: em nenhum dos problemas examinados por FOSTER & RYAN (1976) foi necessário mais do que um corte simples para manter a integralidade da solução. Os planos de corte utilizado pelos autores são da forma:

$$\sum_{r \in R} a_{C_r} x_r \leq \left\lceil \frac{k}{2} \right\rceil$$

onde C é um conjunto de k pontos de demanda e

$$a_{C_r} = \begin{cases} 1 & \text{se a r-ésima rota contém pelo menos dois pontos de C;} \\ \emptyset & \text{c.c.} \end{cases}$$

Um outro algoritmo também baseado em **(PPV3)** é apresentado por RAO & ZIONTS (1968). Os autores utilizam um esquema de geração de colunas onde as rotas são geradas por meio do algoritmo **"Out-of-Kilter"** de fluxo de custo mínimo. O **PPV** tratado por eles é um problema de percursos de navios em um conjunto de portos.

A determinação da variável do Simplex que entra na base envolve a solução de m (um para cada navio) subproblemas **"Out-of-Kilter"**. Uma rota de custo mínimo para cada navio é encontrada e aquela que se mostrar mais

econômica em relação à solução corrente é utilizada para melhorar a solução básica (ou seja, a coluna que entra na base). A solução básica é considerada ótima se nenhum dos navios tem uma rota que torne a solução corrente mais econômica.

Um algoritmo mais recente que parece obter bons resultados é desenvolvido por DESROSIERS et al. (1984). Os autores resolvem um **m-PCV** com restrições de "Time Window", onde os subproblemas são Problemas de Caminho mais Curto com restrições de "Time Window". Estes subproblemas são eficientemente resolvidos por um algoritmo de Programação Dinâmica desenvolvido por DESROSIERS et al. (1983). Este algoritmo é uma extensão do clássico algoritmo de BELLMAN (1958).

II.5 - Algoritmos Baseados na Formulação de Fluxo de Veículos

Nesta formulação usamos variáveis binárias para indicar quando um veículo viaja entre dois nós (pontos de demanda) na solução ótima. Podemos distinguir duas famílias dentro desta formulação:

(i) a que utiliza três índices;

(ii) e a que utiliza dois índices.

No primeiro caso, os três índices dados a cada variável de fluxo indicam a origem do veículo, seu destino e o próprio veículo. No segundo caso o veículo não é identificado.

II.5.1 - Formulações utilizando três índices

LAPORTE & NOBERT (1985) apresentam uma formulação para o PPV-assimétrico involvendo no máximo m veículos e r depósitos. Esta formulação foi baseado no trabalho de GOLDEN et al. (1977) onde são desenvolvidos algoritmos heurísticoos de solução. Esta parece ser uma formulação clássica, apesar de não ter sido desenvolvido nenhum algoritmo exato baseado nela.

Afim de apresentarmos esta formulação, considere a seguinte notação, além da empregada até aqui:

D_k capacidade do veículo k;

L_k o comprimento máximo permitido para a rota k;

δ_i^k o tempo para o veículo k servir o cliente i;

t_{ij}^k o tempo de viajem entre i e j do veículo k;

c_{ij}^k o custo de se usar o veículo k no arco (i,j);

$$x_{ij}^{k} = \begin{cases} 1 & \text{se o veículo } k \text{ viaja de } i \text{ para } j \\ \emptyset & \text{c.c.} \end{cases}$$

Então, o PPV pode ser formulado como:

$$\textbf{(PPV4)} \quad \text{Min} \sum_{i\in N} \sum_{j\in N} \sum_{k=1,m} c_{ij} x_{ij}^{k} \qquad (II.47)$$

sujeito a:

$$\sum_{i\in N'} \sum_{k=1,m} x_{ij}^{k} = 1 \quad (j= r+1,\ldots,n) \qquad (II.48)$$

$$\sum_{j\in N'} \sum_{k=1,m} x_{ij}^{k} = 1 \quad (i= r+1,\ldots,n) \qquad (II.49)$$

$$\sum_{i\in N'} x_{i\ell}^{k} - \sum_{j\in N'} x_{\ell j}^{k} = \emptyset \quad (k= 1,m;\ \ell= 2,n) \qquad (II.50)$$

$$\sum_{i\in N'} d_i \Big(\sum_{j\in N'} x_{ij}^{k} \Big) \leq D_k \quad (k= 1,m) \qquad (II.51)$$

$$\sum_{i\in N'} \delta_i^{k} \sum_{j\in N'} x_{ij}^{k} + \sum_{i\in N} \sum_{j\in N} t_{ij} x_{ij}^{k} \leq L \quad (k= 1,m) \qquad (II.52)$$

$$\sum_{i=1,r} \sum_{j=r+1,n} x_{ij}^{k} \leq 1 \quad (k= 1,\ldots,m) \qquad (II.53)$$

$$\sum_{j=1,r} \sum_{i=r+1,n} x_{ij}^{k} \leq 1 \quad (k= 1,\ldots,m) \qquad (II.54)$$

$$\sum_{i\in S} \sum_{j\notin S} x_{ij}^{k} \geq 1 \quad (|S|\geq 1;\ S\subset N;\ k= 1,m) \qquad (II.55)$$

$$x_{ij}^{k} = \{\emptyset, 1\} \quad (i,j\in N';\ k= 1,m) \qquad (II.56)$$

Nesta formulação as restrições (II.48) e (II.49) especificam que cada cliente deve ser servido exatamente uma vez e por um, e somente um, veículo. As restrições (II.50) garantem a conservação do fluxo de veículos em cada nó da rede. (II.51) e (II.52) são as restrições de capacidade e distância, respectivamente. (II.53) e (II.54) garantem que não mais que m veículos partem do depósito e que existe no máximo um veículo por rota. Finalmente a restrição (II.55) elimina as subrotas ilegais, enquanto que (II.56) é a restrição de integralidade das variáveis.

FISHER & JAIKUMAR (1981) desenvolveram dois algoritmos baseados numa formulação diferente de **(PPV4)**. O primeiro garante uma solução exata enquanto que o segundo utiliza uma heurística eficiente para encontrar uma solução aproximada em um tempo computacional reduzido. Nesta seção trataremos do algoritmo exato, deixando o outro para o próximo capítulo. A formulação é a seguinte:

$$\textbf{(PPV5) Min} \sum_{i\in N} \sum_{j\in N} \sum_{k=1,m} c_{ij} x_{ij}^{k} \qquad \text{(II.57)}$$

sujeito a:

$$\sum_{i\in N'} d_i \, y_{ik} \leq D_k \qquad (k= 1,\ldots,m) \qquad \text{(II.58)}$$

$$\sum_{k=1,m} y_{ik} = \begin{cases} m & \text{se } i= 1 \\ 1 & \text{se } i= 2,\ldots,n \end{cases} \qquad \text{(II.59)}$$

$$\sum_{i\in N'} x_{ij}^{k} = y_{jk} \qquad (j=1,\ldots,n;\ k=1,\ldots,m) \qquad \text{(II.60)}$$

$$\sum_{j \in N'} x_{ij}^{k} = y_{ik} \qquad (i=1,\ldots,n;\ k=1,\ldots,m) \qquad (II.61)$$

$$\sum_{i,j \in S} x_{ij}^{k} \leq |S|-1 \qquad (S \subseteq N;\ 2 \leq |S| \leq n-1;\ k=1,\ldots,m) \qquad (II.62)$$

$$y_{ik} \in \{0,1\} \qquad (i=1,\ldots,n;\ k=1,\ldots,m) \qquad (II.63)$$

$$x_{ij}^{k} \in \{0,1\} \qquad (i,j=1,\ldots,n;\ k=1,\ldots,m) \qquad (II.64)$$

onde: $y_{ik} = \begin{cases} 1 & \text{se a cidade } i \text{ é servida pelo veículo } k; \\ 0 & \text{caso contrário.} \end{cases}$

$x_{ij}^{k} = \begin{cases} 1 & \text{se o veículo } k \text{ trafega no arco } i,j; \\ 0 & \text{caso contrário.} \end{cases}$

Dois conhecidos problemas de otimização combinatória estão presentes na formulação acima:

(i) O problema de Designação Generalizado, dado pelas restrições (II.58), (II.59) e (II.63);

(ii) O Problema do Caixeiro Viajante: quando as variáveis y_{ik} são fixadas de maneira a satisfazer as restrições do Problema de Alocação, as restrições (II.60)-(II.62) e (II.64) definem um PCV para um dado veículo k.

Os autores desenvolvem um algoritmo baseado na

decomposição de Benders para resolver o **(PPV5)** dado pela formulação acima. A idéia básica do método pode ser descrita através da seguinte reformulação do **PPV** em termos do Problema de Alocação (não linear):

$$\text{Min} \quad \sum_{k} f(y_k) \tag{II.65}$$

sujeito a: (II.58),(II.59) e (II.63)

onde $f(y_k)$ é o custo de uma rota ótima do PCV dos clientes pertencentes a $N(y_k) = \{i : y_{ik} = 1\}$.

A função $f(y_k)$ pode ser definida matematicamente, para cada k, por:

$$f(y_k) = \text{Min} \sum_{i,j,k} c_{ij} x_{ij}^{k} \tag{II.66}$$

sujeito a: (II.60)-(II.62) e (II.64)

A cada passo o Problema de Alocação é resolvido, uma cota inferior para $f(y_k)$ é calculada a partir das variáveis duais do problema acima e adicionado ao Problema de Alocação. Para obter as variáveis duais, a integralidade em x_{ij}^{k} é relaxada ou é imposta através de cortes padrões de Programação Inteira.