# O PROBLEMA DO CAIXEIRO VIAJANTE (PCV): UM ENFOQUE PARA O SISTEMA DE MANUFATURA FLEXÍVEL (SMF)

Miriam Izu

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 (M.Sc.) EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Dina Feigenbaum Cleiman (D.Sc.)
Presidente

Carlos Alberto Nunes Cosenza (D.Sc.)

Luis Satoru Ochi (D.Sc.)

RIO DE JANEIRO, RJ – BRASIL
DEZEMBRO DE 1991

**Izu, Miriam**

O Problema do Caixeiro Viajant (PCV): Um Enfoque para o Sistema de Manufatura Flexível (SMF) (Rio de Janeiro), 1991

viii, 92 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 1991)

Tese – Universidade Federal do Rio de Janeiro, COPPE.

1. Otimização

I. COPPE/UFRJ II. Título (Série)

A meus pais, Milton e Teresa
pelo incentivo e carinho.
A meus irmãos e a você Dudu
que é sempre motivo de alegria.

## AGRADECIMENTOS

A Professora Dina F. Cleiman, pela orientação, incentivo e todo o apoio dado, tornando possível a realização deste trabalho.

Ao Professor Luiz Satoru Ochi que vem nos incentivando desde os tempos da UFF, pelas sugestões importantes dadas.

A CAPES pela bolsa concedida.

Aos amigos, principalmente o Coca pelas dúvidas tiradas e outras tantas colocadas e a Ingrid, que desde a UFF é uma amiga que sempre pode–se contar, pelo incentivo.

Aos Professores, colegas e funcionários da COPPE, pela orientação, convívio agradável e todo apoio recebidos.

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

**O PROBLEMA DO CAIXEIRO VIAJANTE (PCV): UM ENFOQUE PARA O SISTEMA DE MANUFATURA FLEXÍVEL (SMF)**

**Miriam Izu**

Dezembro de 1991

**Orientadora:** Dina Feigenbaum Cleiman

**Programa:** Engenharia de Sistemas e Computação

São apresentadas algumas formulações do PCV e os principais métodos de resolução exatos e heurísticos. Descreve–se alguns problemas de SMF e a sua resolução através do Problema do Caixeiro Viajante.

Abstract of thesis presented to COPPE as partial fulfillment of the requeriments for the degree of Master of Science (M. Sc.)

**THE TRAVELING SALESMAN PROBLEM (TSP): AN APPROACH TO THE FLEXIBLE MANUFACTURING SYSTEM (FMS)**

**Miriam Izu**

Dezembro, 1991

**Thesis Supervisor:** Dina Feigenbaum Cleiman

**Department:** Systems Engineering and Computing

We presented some formulations of the traveling salesman problem and the main solution methods exacts and heuristics. Problems of the FMS are modeling and for its solutions are used traveling salesman's algorithms

# ÍNDICE

# CAPÍTULO I

## INTRODUÇÃO

Um dos problemas mais conhecidos e estudados em Otimização Combinatória é o Problema do Caixeiro Viajante (PCV). Este problema busca, em sua forma padrão, descobrir uma seqüência de visitação a uma dada lista de cidades. Conhece–se a distância entre cada par de cidades e o caixeiro deve visitar cada uma delas exatamente uma vez retornando então à cidade inicial de maneira tal que a distância percorrida seja mínima.

À despeito da facilidade de apresentação e formulação, o PCV apresenta uma dificuldade de resolução que o coloca na categoria dos problemas conhecidos como NP–difíceis [JOHNSON e PAPADIMITRIOU, 1985].

O PCV apesar de ser, de maneira geral, de difícil resolução apresenta alguns casos particulares que podem ser eficiente e facilmente resolvidos. Alguns desses casos são: PCV com matriz de distâncias C triangular superior; PCV constante (todos os circuitos têm a mesma distância), PCV com matrizes "pequenas" [GILMORE, LAWLER e SHMOYS, 1985].

Além de sua reconhecida importância teórica, o PCV tem uma grande aplicação prática. Diversos problemas práticos podem ser formulados como instâncias do PCV. Dentre estes problemas podemos destacar: roteamento de veículos, programação de vôos e tripulações, planejamento de conexões elétricas em computadores.

Neste trabalho procuramos apresentar uma aplicação do PCV relacionado ao problema de seqüenciamento cíclico de tarefas em m–máquinas. Na primeira parte do trabalho apresentamos um estudo dos modelos e métodos de resolução do PCV. Na segunda parte da dissertação apresentamos um Sistema de Manufatura Flexível (SMF) a partir de alguns tipos de problemas de manufatura flexível.

Neste primeiro capítulo definimos o PCV padrão, classificamos os tipos de PCV e os principais métodos de resolução. No Capítulo II desenvolvemos os principais métodos exatos de resolução e no Capítulo III os principais métodos heurísticos. No Capítulo IV introduzimos o conceito de manufatura flexível e alguns problemas de SMF. Finalmente, no Capítulo V são apresentadas conclusões que chegamos com este estudo.

## I.1 – DEFINIÇÃO E CLASSIFICAÇÃO DO PCV BÁSICO

**Definições:**

Seja $G = (V, A)$ um grafo completo orientado ou não, onde V é um conjunto de n vértices e A um conjunto de m arcos (ou arestas se o grafo for não orientado).

Seja $c_{ij}$ o comprimento do arco (i, j), $c_{ij}$ dependendo do modelo considerado pode ser custo, tempo, etc. Cada arco de A é um par ordenado de vértices $\ell = (i, j)$ onde i é a extremidade inicial e j a extremidade final do ano $\ell$.

Um **caminho** é uma seqüência de arcos $\{\ell_1, ..., \ell_p\}$ tal que para todo $k = 1, ..., p-1$ a extremidade final de $\ell_k$ coincide com a extremidade inicial de $\ell_{k+1}$.

Um **circuito** é um caminho $\{\ell_1, ..., \ell_p\}$ para o qual a extremidade final do ano $\ell_p$ coincide com a extremidade inicial do ano $\ell_1$.

Um circuito é dito ser um **circuito elementar** se:

i) há no máximo um arco tendo o vértice i como extremidade inicial para cada vértice i;

ii) há no máximo um arco tendo o vértice i como extremidade final para cada vértice i.

Um **caminho hamiltoniano** é um circuito elementar que passa por todos os vértices do grafo.

Podemos definir então o PCV como a busca do circuito hamiltoniano de comprimento mínimo no grafo G.

O problema de determinar se um dado grafo possui um circuito hamiltoniano é NP completo [NEMHAUSER e WOLSEY, 1988].

PADBERG e SUNG (1991) apresentam quatro formulações diferentes para o PCV como veremos a seguir.

DANTZIG, FULKERSON e JOHNSON em um paper publicado em 1954 formularam o problema como um modelo de programação linear 0–1 envolvendo $0(n^2)$ variáveis e $0(n^2)$ restrições lineares. Como a formulação por eles apresentada envolve um número exponencial de restrições, diversos outros pesquisadores propuseram formulações que envolvem um número polinomial de restrições, embora isto tenha levado a um aumento do número de variáveis. Apresentaremos aqui as formulações de MILLER, TUCKER e ZEMLIN (1960); FOX, GAVISH e GRAVES (1980) e CLAUS

(1984).

**Formulação de DANTZIG–FULKERSON e JOHNSON (DFJ)**

O PCV foi formulado como um problema de Programação Linear 0–1 sobre um grafo G = (V, A), como definido anteriormente:

$$\min \sum_{i,j=1}^{n} c_{ij}\, x_{ij}$$

s.a.

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, \ldots, n \tag{I.1}$$

$$\sum_{i=1}^{n} x_{ji} = 1, \quad j = 1, \ldots, n \tag{I.2}$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - 1 \quad \forall S \subseteq V \text{ e } 2 \leq |S| \leq n-1 \tag{I.3}$$

$$x_{ij} \geq 0 \quad \forall i,j = 1, \ldots, n \tag{I.4}$$

$$x_{ij} \text{ inteiro } \forall i,j = 1, \ldots, n \tag{I.5}$$

Vamos assumir em todo este estudo que $x_{ii} = +\infty$, $\forall i = 1, \ldots, n$ e que

$$x_{ij} = \begin{cases} 1, \text{ se o caixeiro viaja diretamente da cidade i para a cidade j} \\ 0, \text{ c/c (caso contrário)} \end{cases}$$

Esta formulação envolve então n(n–1) variáveis 0–1 e $0(2^n)$ restrições. A restrição (I.3) é uma restrição de eliminação de subrotas e não admite circuitos visitando um subconjunto de nós em V.

**Formulação de Miller–Tucker–Zemlim (MTZ)**

Foi apresentada em 1960 e propõe uma formulação mais geral para o PVC sobre $V = \{1, ..., n\}$ nós.

Seja a cidade inicial o nó 1. Deseja–se que o caixeiro visite as restantes $(n-1)$ cidades exatamente uma vez. Durante sua viagem ele deve voltar t vezes à cidade inicial, incluido o último retorno e não pode visitar mais do que p cidades diferentes da 1 em uma rota.

**Definição:**

Definimos aqui por **rota** a sucessão de visitas a cidades sem parar na cidade inicial (1). Esta, todavia, não seria a definição de rota com que trabalharemos em todo este estudo, salvo alguma referência específica. Esta definição de rota prende–se à características próprias desta formulação.

É requerido que:

i) $\lceil (n-1)/p \rceil \leq t \leq n-1$, onde

ii) $p \geq 2$

$\lceil a \rceil$ denota o menor inteiro maior ou igual que a, para $\forall\, a \in \mathbb{R}$.

O item i) assegura rotas viávris e ii) elimina o caso trivial.

Escrevendo como um PL 0–1 misto tem–se:

$$\text{minimizar} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}\, x_{ij}$$

sujeito a

$$\sum_{i=2}^{n} x_{i1} = t \tag{I.6}$$

$$\sum_{i=2}^{n} x_{1i} = t \tag{I.7}$$

$$\sum_{i=2}^{n} x_{ij} = 1, \quad j = 2, \ldots, n \tag{I.8}$$

$$\sum_{j=2}^{n} x_{ij} = 1, \quad i = 2, \ldots, n \tag{I.9}$$

$$u_i - u_j + p\, x_{ij} \leq p-1, \qquad 2 \leq i \neq j \leq n \tag{I.10}$$

$$u_i \geq 0, \qquad 2 \leq i \leq n \tag{I.11}$$

$$n_{ij} \geq 0 \quad \forall\, i,j \tag{I.12}$$

$$x_{ij} \text{ inteiro } \forall\, i,j \text{ e } u \in \mathbb{R}^n \tag{I.13}$$

Com as restrições (I.6) e (I.7) nós garantimos que a cidade inicial é visitada exatamente t vezes. A restrição (I.10) elimina subrotas que não contém a origem.

Esta formulação usa n–1 variáveis e $O(n^2)$ restrições. Para $t = 1$ e $p \geq n-1$ a formulação MTZ modela o PCV padrão corretamente.

**Formulação de Fox–Gavish–Graves (FGG)**

Foi formulada em 1980 para um PCV com "restrições de tempo" (time dependent).

Nesta formulação levamos em consideração a posição t do arco (i,j) na rota relativa à primeira cidade (cidade de origem) cujo índice é 1 para determinar o custo de se viajar de ir para j.

O PCV com restrições de tempo foi originalmente proposto como uma formulação para o problema de scheduling de n tarefas em uma máquina, o que pode ser referenciado em FOX (1973), PICARD e QUEYRANNE (1978).

As variáveis de decisão serão de três índices $x_{ijt}$ significando:

$$x_{ijt} = \begin{cases} 1 & \text{se o arco de i para j esta alocado na t–ésima posição da rota} \\ 0 & \text{caso contrário} \end{cases}$$

Tem–se então o problema:

$$\text{minimizar} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{t=1}^{n} c_{ijt}\, x_{ijt}$$

sujeito a

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{t=1}^{n} x_{ijt} = n \qquad (I.14)$$

$$\sum_{j=1}^{n} \sum_{t=2}^{n} t\, x_{ijt} - \sum_{j=1}^{n} \sum_{t=1}^{n} t\, x_{jit} = 1, \quad i = 2, \ldots, n \qquad (I.15)$$

$$x_{ijt} \geq 0 \quad \forall\, i,j,t \qquad (I.16)$$

$$x_{ijt} \text{ inteiro} \quad \forall,\, i,j,t \qquad (I.17)$$

$t \geq 0$ inteiro

A restrição (I.14) é a "agregação" de restrições de um Problema da Designação 3–dimensional o que assegura que cada cidade é visitada exatamente uma vez e que posição da rota tem um arco designado para ela.

A restrição (I.15) assegura que para cada cidade diferente da cidade inicial a posição de um arco saindo desta cidade é exatamente uma unidade a mais do que a posição de um arco entrando. Para uma cidade em uma subrota de comprimento $K = 2$ a posição de um arco deixando a cidade é K unidades maior do que a posição de um arco entrando na cidade e portanto a restrição (I.15) não pode ser satisfeita a menos que a cidade seja a cidade inicial. Segue–se que cada solução viável 0–1 corresponde a uma rota e que a restrição (I.15) elimina todas as subrotas que não contém a origem.

Podemos reescrever as restrições (I.15) de forma:

$$-\sum_{j=1}^{n} x_{ji1} + n(x_{ijn} - x_{jin}) + \sum_{t=2}^{n-1} t \sum_{j=1}^{n} (x_{ijt} - x_{jit}) = 1$$

para $i = 2, ..., n$.

Observa–se que:

$$x_{ij1} = x_{jin} = x_{1it} = x_{iit} = 0$$

para todo $i \geq 2$, $j \geq 1$ e $2 \leq t \leq n-1$ em cada solução viável para (I.14)–(I.17), estas restrições são simplicadas para:

$$-x_{1i1}+nx_{iin} + \sum_{t=2}^{n-1} t \sum_{j=2}^{n} (x_{ijt}-x_{jit}) = 1, \quad \forall i=2,...,n \qquad (I.18)$$

As variáveis do tipo $x_{iit}$ para todo i e t não são necessárias bem como aquelas que assumem o valor zero em cada solução viável de (I.14)–(I.17) sendo portanto abandonadas no modelo.

No caso em que $c_{ijt} = c_{ij}$ $\forall$ i,j e t, a formulação FGG modela o PCV padrão e envolve n restrições lineares de m variáveis 0–1 sendo

$$m = 2(n-1) + (n-1) \cdot (n-2)^2$$

**Formulação de Claus (C)**

Foi proposta por Claus em 1984 e usa o conceito de fluxos em redes envolvendo diversos produtos. Denotemos por s a cidade inicial (a "origem") e transformemos qualquer circuito hamiltoniano em um caminho hamiltoniano pela duplicação da cidade inicial em outra cidade t (sumidouro). O PCV pode ser interpretado como o problema de achar um caminho hamiltoniano de s para t sobre um $(s, t)$–dígrafo $G = (V, A)$ onde $V = V^1 \cup \{s, t\}$, $V^1 = \{1, ..., n\}$ e $A = \{(i, j) | \forall i \neq j \in V^1] \cup \{(s, i), (i, t) | \forall i \in V^1\}$ e a variável de decisão

$$x_{ij} = \begin{cases} 1 & \text{se } (i,j) \text{ está no caminho hamiltoniano} \\ 0 & \text{caso contrário} \end{cases}$$

define a capacidade de um arco (i,j). O fluxo da rede envolve n+1 produtos. O "k–ésimo produto" é produto enviado de s para o vértice k onde k = 1, ..., n+1 e n+1 é o índice do sumidouro t.

Seja a variável $y_{ijk}$ o fluxo do k–ésimo produto no arco (i,j). Claus formulou como o seguinte modelo linear 0–1:

$$\text{minimizar} \sum_{(i,j) \in A} c_{ij} x_{ij}$$

sujeito a

$$\sum_{j \in V} x_{ij} = \sum_{j \in V} x_{ji} = 1 \quad \forall i \in V^1 \tag{I.19}$$

$$\sum_{i \in V^{\prime}} x_{si} = \sum_{i \in V^{\prime}} x_{it} = 1 \tag{I.20}$$

$$\sum_{j \in V} y_{ijk} - \sum_{j \in V} y_{jik} = 0 \quad \forall i \in V^{\prime}, K \in V^{*} \tag{I.21}$$

$$\sum_{i} - y_{ikk} = -1, \sum_{i} y_{sik} = 1, \sum_{i} y_{kik} = 0, \quad \forall k \in V^{*} \tag{I.22}$$

$$y_{ijk} \leq x_{ij} \quad \forall i,j,k \tag{I.23}$$

$x_{ij} \geq 0 \quad \forall\, i,j$ (I.24)

$y_{ijk} \geq 0 \quad \forall\, i,j,k$ (I.25)

$x_{ij}$, $y_{ijk}$ inteiros $\forall\, i,j,k$ (I.26)

onde $V^{*} = V' \cup \{t\}$.

As restrições (I.21), (I.22) e (I.25) asseguram que para qualquer nó $k \in V^{*}$ existe um **caminho** da origem s para o nó k.

A restrição (I.23) une estes requerimentos através de todos os nós exigindo adicionalmente que para todos os vetores viáveis (x, y) tal caminho existe no **grafo suporte** correspondente para valores positivos de x.

Segue–se que no grafo suporte cada corte tem um valor de ao menos um e portanto todas as restrições de eliminação de subrotas são satisfeitas para valores viáveis de x.

Para cada solução viável de (I.18)–(i.24) nós temos $y_{itk} = 0$, $\forall\, i \in V$ e $K \in V'$ porque nenhum produto pode ser enviado do sumidouro t e $y_{kik} = 0 \quad \forall\, i,k \in V^{*}$ porque $\sum_{i} y_{kik} = 0 \quad \forall\, k \in V^{*}$. Podemos então descartar estas variáveis da formulação.

A formulação de Claus envolve $n^3 + n^2 + 3n$ variáveis e um conjunto de restrições da ordem de $0(n^3)$.

**Formulação de FINKE–CLAUS–GUNN (FCG)**

Seja $G = (V, A)$ um grafo direcionado onde V é um conjunto de n vértices e A um conjunto de m arcos com $c_{ij}$ representando os custos dos arcos $(i, j) \in A$.

Q e R são dois produtos distintos.

Selecionando um vértice s qualquer de V, podemos definir as quantidades $q_i$ de um produto Q e $r_i$ de um produto r da seguinte maneira em cada no i

$$q_i = \begin{cases} n-1 & \text{se } i = s \\ -1 & \text{caso contrário} \end{cases}$$

e

$$r_i = \begin{cases} -(n-1) & \text{se } i = s \\ 1 & \text{caso contrário} \end{cases}$$

Consideremos que $x_{ij}$ e $y_{ij}$ representam os fluxos de Q e R respectivamente que saem de i e chegam em j.

As quantidades $q_i$ e $r_i$ indicam que cada nó oferta uma unidade do produto Q e demanda uma unidade do produto R, sendo que o produto Q tem uma oferta inicial de n unidades e produto R terá oferta nula. Portanto números positivos indicam oferta e números negativos demandas. Suponhamos que o caixeiro viajante inicie o circuito hamiltoniano no vértice s. Ele carrega de s as n unidades do produto Q. No próximo vértice da rota, ele deixa uma unidade de Q e pega uma unidade de R, mantendo–se o total de (n–1) unidades no próximo arco, consistindo de (n–2) unidades de Q e uma unidade de R.

A troca de uma unidade de cada produto continua até que finalmente (n−1) unidades de R chegam em s.

O problema então é formulado como:

$$\text{minimizar } \frac{1}{n-1} \sum_{i,j} c_{ij}(x_{ij} + y_{ij})$$

sujeito a:

$$\sum_{j \in V} x_{ij} - \sum_{j \in V} x_{ji} = q_i \qquad \forall i \in V \tag{II.27}$$

$$\sum_{j \in V} y_{ij} - \sum_{j \in V} y_{ji} = r_i, \qquad \forall i \in V \tag{II.28}$$

$$\sum_{j \in V} (x_{ij} + y_{ij}) = n-1, \;\; \forall i \in V \tag{II.29}$$

$$x_{ij} + y_{ij} \in \{0, n-1\}, \quad \forall\, (i,j) \in A \tag{II.30}$$

$$x_{ij} \geq 0, \; y_{ij} \geq 0, \quad \forall\, (i,j) \in A \tag{II.31}$$

As restrições (I.27), (I.28) e (I.31) definem um fluxo de rede viável em cada nó i. As soluções (I.29) e (I.30) asseguram que há exatamente um arco deixando cada vértice i $\in$ V que leva um fluxo total de (n−1) unidades, pois (I.29) obriga que a soma dos fluxos de Q e R, que saem de cada nó seja igual a n−1 e a restrição (I.30) determina que a soma dos fluxos de Q e R seja O ou n−1. Garante−se então que apenas um arco que sai de i poderá ter um fluxo de n−1 unidades, os restantes terão fluxo zero (arcos não considerados na solução). Portanto, de cada vértice sai apenas um arco.

Logo as restrições (I.27) a (I.31) caracterizam um circuito hamiltoniano, devendo esta formulação apresentar uma fundamental importância nas restrições de precedência que serão consideradas ao modelarmos um problema de SMF no Capítulo IV.

O PCV pode ser classificado em relação à sua matriz de custos $C = (c_{ij})$ como sendo **simétrico** se a matriz C é simétrica e **assimétrica** caso contrário. Se a matriz C satisfaz a desigualdade triangular o problema é dito **Euclidiano**, ou seja se $c_{ij} \leq c_{ik} + c_{kj} \quad \forall i, j \; k \in N$ e $i \neq j \neq k$. Caso isto não aconteça o problema é dito **não–euclidiano.**

Se o problema é Euclidiano cada elemento da matriz é a menor distância entre duas cidades.

A suposição de obediência à desigualdade triangular permite especificar que cada cidade seja visitada uma única vez na solução ótima e é a formulação mais encontrada na literatura.

Apresentamos nesta dissertação uma aplicação do PCV, o scheduling de tarefas, onde a desigualdade triangular não é satisfeita, permitindo–se assim que alguns vértices sejam visitadas mais de uma vez para se conseguir a rota de custo mínimo.

## I.2 – CLASSIFICAÇÃO DOS MÉTODOS DE RESOLUÇÃO

Para resolução do PCV a literatura apresenta basicamente, dois tipos de métodos: os exatos e os aproximados.

Ao nos depararmos com problemas do tipo NP–árduo, como é o caso do PCV, não temos garantia de conseguir uma solução ótima em uma quantidade razoável de tempo. Podemos escolher então entre dois tipos de estratégias: insistir na otimalidade da solução (com risco de grande perda de tempo) ou usar um método de solução rápida (aceitando soluções quase ótimas [HOFFMAN e WOLFE, 1985].

Na prática, a estratégia de aproximação é a mais utilizada.

Com os métodos exatos obtemos uma solução ótima para o problema utilizando técnicas de Programação Matemática. Como o PCV é NP–difícil, sua aplicação so é possível em problemas de pequeno porte.

Programação Dinâmica também foi aplicada para o PCV (BELLMAN, 1962; HELD e KARP, 1962]. A programação dinâmica pode resolver apenas instâncias de problemas relativamente pequenos, devido à sua necessidade de armazenar elevado número de dados.

Ao optarmos por um método aproximado utilizamos a estrutura do problema dado aplicando uma heuristica que obtem uma solução viável "quase ótima" do problema. Esta solução viável "quase ótima" é uma solução que satisfaz as restrições do problema e fornece um valor para a função objetivo bastante próximo do ótimo.

# CAPÍTULO II

# MÉTODOS EXATOS

Veremos neste capítulo os principais métodos exatos para resolução do PCV.

## II.1 – PLANOS DE CORTE

O primeiro algoritmo de planos de corte foi desenvolvido por Gomory em 1958 para o problema inteiro puro. Glover, em 1965, introduziu outros tipos de cortes assim como YOUNG (1971), BALAS (1971).

A técnica de planos de corte ou simplesmente corte é usada para Programas Lineares Inteiros (PLI), que podem ser formulados como:

$$(PLI)\begin{cases} \min\ z = cx \\ s.a. \\ A.x = b \\ x \geq 0,\ \ x_j \text{ inteiro } \forall j = 1, ..., n \end{cases}$$

Utilizaremos o método simplex no Programa Linear (PL) obtido do PLI ao se relaxar a condição de integrabilidade. Se a solução ótima obtida for inteira o problema está resolvido.

Se a solução obtida não for inteira acrescentaremos um plano de corte ou simplesmente corte que é uma restrição linear gerada à partir da solução ótima atual com as seguintes características: elimina a solução ótima não inteira do problema linear atual sem excluir nenhuma solução inteira viável do problema original. Usamos o simplex para resolver o problema acrescido do corte, se a solução for inteira o problema está terminado. Caso contrário acrescentamos um novo corte com as propriedades anteriores, resolvemos novamente o problema e assim por diante.

Este procedimento termina quando o simplex deteta inviabilidade, ilimitabilidade ou otimalidade (inteira).

A escolha de cortes convenientes faz com que o poliedro P, que representa o PL, vá diminuindo até que coincida com a envoltória convexa das soluções inteiras, no mínimo nas vizinhança da solução ótima. Teremos então que a solução contínua do problema linear se torna inteira resolvendo–se o problema MINOUX (1986).

Infelizmente, não há garantia de que com sucessivo acréscimo de cortes iremos obter uma solução inteira em um número finito de iterações TAHA (1975).

Desse modo, a maneira como os cortes são gerados irá afetar profundamente a eficácia do método.

Destacaremos os planos de cortes de Gomory que têm convergência finita garantida MINOUX (1986).

Uma desvantagem que podemos notar no método de planos de corte é que não podemos contar com uma solução até o término do algoritmo, já que na sua maioria eles são do tipo dual.

## II.2 – BRANCH AND BOUND – PARTICIONAR E LIMITAR

Dentre os métodos exatos que se destacam para a resolução de problemas de Otimização Combinatória, branch and bound, sem dúvida, é o mais utilizado.

Os primeiros a proporem o métodos para Problemas Lineares Inteiros Mistos (PLIM) foram Land e Doig em 1960 [GARFINKEL]. Vários outros pesquisadores também desenvolveram diversos trabalhos sobre este assunto, dentre eles: TOMLIM (1970); GEOFFRION e MARSTEN (1972); LAND e POWELL (1977).

Os métodos "branch and bound" que foram desenvolvidos especialmente para Problemas Lineares Inteiros (PLI) binários por Balas, Geoffrion e Glover, dentre outros, são denominados "enumeração implícita" por apresentarem características próprias relevantes, que os distinguem do "branch and bound"

O método "branch and bound" tem basicamente duas rotinas principais que são, como o próprio nome indica, particionar e limitar.

No processo de particionamento ("branching") dividimos o conjunto de soluções viáveis em subconjuntos cada vez menores que estão associados a subproblemas. Estes subconjuntos têm a particularidade de possuir qualquer solução viável em ao menos um destes subconjuntos.

Então, se o problema que temos é da forma:

(P) $\min z = cx$

s.a.

$Ax \leq b$

$x \geq 0$

$x_j$ inteiro, $j \in I$

onde as matrizes c, A e b são de dimensão (1 x n), (m x n) e (m x 1) respectivamente e x tem dimensão (n x 1).

**Notação**: S é o conjunto de soluções viáveis de (P)

$S = \{x/Ax \leq b,\ x \geq 0,\ x_j \text{ inteiro},\ j \in I\}$

Ao realizarmos a partição, a regra utilizada deverá garantir que ao particionarmos o conjunto viável $S_i$ do problema corrente $P_i$ em subconjuntos $S_{i_1}, \ldots, S_{i_k}$ teremos $\bigcup_{j=1,k} S_{i_j} = S_i$.

Resolvendo cada subproblema gerado, obteremos com o valor objetivo ótimo deste, um limite inferior (superior) para um problema original que era do tipo minimizar (maximizar). Estes limitantes serão usados para descartar soluções que não são promissoras (não melhoram o valor do ótimo obtido até então) ou selecionar subconjuntos que têm possibilidade de conter uma solução ótima.

Os limites do procedimento "limitação" (bounding) são obtidos pela resolução de problemas **relaxados**, que são conseguidos descartando-se as restrições "difíceis" do problema. Estes problemas relaxados são mais fáceis de resolver e o valor da sua solução ótima limita o valor da solução do problema original.

Terminaremos este procedimento quando cada subconjunto (subproblema) tiver sido resolvido. A resolução de um subproblema para em um dos casos abaixo:

i) o subproblema relaxado é inviável;

ii) a solução ótima do subproblema relaxado é inviável para o problema original. Este subproblema será particionado se o valor desta solução ótima for menor que o valor da melhor solução viável armazenada e incluiremos os subproblemas obtidos na lista de subproblemas a resolver;

iii) a solução ótima do subproblema relaxado é viável para o problema original. Se este valor da solução ótima é menor do que o valor da melhor solução viável, esta é atualizada.

A solução obtida que tem o melhor valor (a menor solução) é uma solução ótima global.

Os subproblemas gerados são armazenados em uma lista. A escolha do próximo subproblema a ser resolvido apresenta duas regras bastante utilizadas que são:

a) menor limitante inferior;

Escolhe o nó k (subproblema) da árvore de busca que tem o menor limite inferior dentrre os nós que não foram nem descartados nem particionados.

Dentre as desvantagens deste método aparecem dificuldade de armazenagem e se houver término antecipado devido à limitações de tempo poderemos não contar com uma boa solução viável. A vantagem desta regra é a de manter a árvore reduzida, ou seja, tende a reduzir o número de problemas a serem resolvidos.

b) LIFO (Last in First Out)

Escolhe o último subproblema gerado.

Sua armazenagem é mais eficiente e podemos dispor de uma boa solução viável rapidamente. Apresenta o inconveniente de descobrir soluções que poderiam ter sido descartadas com o uso da regra anterior.

Para tirar proveito das duas regras podemos combiná–las usando LIFO para tentar obter uma boa solução viável inicialmente e depois usar a regra do menor limitante inferior para tentar manter a árvore de busca pequena GARFINKEL (1979).

Ao desenvolver um algoritmo do tipo "branch and bound" devemos ter em mente quatro rotinas que são essenciais para a eficácia do método. Se o problema P em mãos for da seguinte forma:

P: $\min f(x)$
s.a. $x \in S$

onde S é o conjunto das soluções viáveis.

As rotinas básicas serão:

i) relaxação de P:

Um problema Q da forma

Q: $\min g(x)$
s.a. $x \in T$

onde T é o conjunto das soluções viáveis tal que $S \subseteq T$ e para cada x, $y \in S$ se

$f(x) < f(y)$ então $g(x) < g(y)$;

ii) "Branching" (separação)

Uma regra que particione o conjunto viável $S_i$ do problema corrente $P_i$ em subconjunto $S_{i_1}, \ldots, S_{i_k}$ tal que $\bigcup_{j=1,\ldots,k} S_{i_j} = S_i$;

iii) Procedimento de limitação inferior

Um procedimento para achar (ou uma aproximação) para a relaxação $Q_i$ de cada subproblema $P_i$.

iv) Regra de seleção de subproblema

Escolhe o próximo subproblema da lista a ser resolvido.

Nos métodos "branch and bound" a qualidade dos limites calculados tem um peso maior na eficácia do algoritmo do que a escolha de qualquer regra de "branching" que possa ser usada para gerar os subproblemas durante a busca [CHRISTOFIDES, 1979].

Diversos algoritmos "branch and bound" distintos podem ser conseguidos pela combinação das diversas estratégias de partição e limitação.

Apresentaremos a seguir algumas destas estratégias.

### II.2.1 – Limites Obtidos da Árvore Geradora Mínima

**PCV Simétrico**

Seja $G = (V, A)$ um grafo completo não direcionado onde V é um conjunto de n vértices e A um conjunto de m arcos e seja r o ciclo hamiltoniano que é a solução do PCV. Seja v um vértice em r, $c_{ij}$ é o custo do arco (i, j). Se removermos v e os dois arcos incidentes em v teremos que o grafo restante forma um caminho, C, passando pelos vértices de $V - \{v\}$. Se $T_v$ é a árvore mínima geradora do grafo $<N - \{v\}>$, temos que o custo de $T_v$ é um limite inferior sobre o custo de C.

O custo dos dois arcos que foram removidos de r para obter P tem um custo total no mínimo tão grande quanto a soma dos dois de menor peso, $(i_1, v)$ e $(i_2, v)$ incidentes em v.

Denotemos por $G_v$ o grafo composto dos arcos em $T_v$ e pelos arcos $(i_1, v)$ e $(i_2, v)$. Claramente, o custo total dos arcos em $G_v$ é um limite inferior sobre o custo de r.

O problema $P_T$ cuja solução pode ser dada pelo grafo $G_v$ pode ser formulado matematicamente como:

$$\text{minimizar} \quad \sum_{\ell=1}^{m} c_\ell x_\ell \tag{II.1}$$

s.a.

$$\sum_{\ell=1}^{m} x_\ell = n \tag{II.2}$$

$$\sum_{\ell \in k_x} x_\ell \geq 1 \quad \forall k_x \equiv (S_k, \bar{S}_k)\ S_x \subset V \text{ e } \bar{S}_x = V - S_t \tag{II.3}$$

PT $\sum_{\ell \in A_V} x_\ell = 2$ $A_V$ é o conjunto de arcos incidentes no vértice V

(II.4)

$$x_\ell \in \{0, 1\}\ \ell = 1, \ldots, m \tag{II.5}$$

Como visto anteriormente, dado o grafo G = (V, A) e os custos $c_{ij}$ de cada arco (i, j), podemos formular o PCV como a busca do circuito Hamiltoniano de custo mínimo:

minimizar $$\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}\, x_{ij} \tag{II.6}$$

s.a.

PCV:

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \forall j \in V \tag{II.7}$$

$$\sum_{j=1}^{n} x_{ij} \quad \forall i \in V \tag{II.8}$$

$$x_{ij} \in \{0,1\} \quad \forall i,j \in V \tag{II.9}$$

$x_{ij}$ deve formar uma rota (II.10)

tendo

$$x_{ij} = \begin{cases} 1 \text{ se } (i,j) \text{ está na rota ótima (circuito hamiltoniano)} \\ 0 \text{ c/c} \end{cases}$$

A restrição (II.10) pode ser escrita de diferentes maneiras, dentre as quais:

$$\sum_{i \in S_t} \sum_{j \in \bar{S}_t} x_{ij} \geq 1 \quad \forall\, S_t \subset V \tag{II.10a}$$

$$\sum_{i \in S_t} \sum_{j \in S_t} x_{ij} \leq |S_t| - 1 \quad \forall\, S_t \subset V \tag{II.10b}$$

$$\sum_{(i,j) \in r} x_{ij} \leq |S_t| - 1 \quad \forall S_t \subset V \text{ e todo } r \in \mathcal{F}(S_t) \tag{II.10c}$$

onde $\mathcal{F}(S_t)$ é a família de todos os circuitos hamiltonianos do subgrafo induzido $\langle S_t \rangle$ definido pelo conjunto de vértices $S_t$. r será usado para representar tanto o ciclo hamiltoniano quanto o conjunto de arcos que formam tal circuito.

$K_t \equiv (S_t, \bar{S}_t)$ é o conjunto de arcos (i, j) com $i \in S_t$ e $j \in \bar{S}_t$. $K_t$ será referido como um conjunto de corte de arcos de G.

A restrição (II.10a) reflete o fato de que ao menos um arco na rota do PCV deva pertencer a algum conjunto de corte de arcos $K_t$ de G. Já as restrições (II.10b) e (II.10c) impedem o aparecimento de subrotas na solução do PCV.

No caso em que G é um grafo não direcionado, seja $\ell$ o índice do conjunto de arcos A e $c_\ell$ o custo do arco $\ell$.

Seja:

$$x_\ell = \begin{cases} 1 \text{ se o arco } \ell \text{ está na solução} \\ 0 \text{ c/c} \end{cases}$$

O PCV simétrico pode ser formulado como:

$$\text{minimizar} \sum_{\ell=1}^{m} c_\ell x_\ell \qquad \text{(II.11)}$$

$$\text{PCV}_s: \qquad \text{s.a.} \sum_{\ell=1}^{m} x_\ell = n \qquad \text{(II.12)}$$

$$\sum_{\ell \in k_t} x_\ell \geq 2 \quad \forall\, k_\ell \equiv (S_t, \bar{S}_t),\ S_t \subset N \qquad \text{(II.13)}$$

$$x_\ell \in \{0,1\} \quad \ell = 1, \ldots, M \qquad \text{(II.14)}$$

Observemos que as restrições (II.12) e (II.13) são equivalentes à:

$$\sum_{\ell \in K_t} x_\ell \geq 1 \quad \forall\, K_t \equiv (S_t, \bar{S}_t),\ S_t \subset V \qquad \text{(II.15a)}$$

e

$$\sum_{\ell \in A_i} x_\ell = 2 \quad i = 1, \ldots, n \qquad \text{(II.15b)}$$

onde $A_i$ é o conjunto de arcos incidentes ao vértice i.

Os problemas $\text{PCV}_S$ e $P_T$ são iguais a não ser pela restrição (II.13) que foi substituída pela (II.15a) e a restrição (II.15b) que for ignorada.

Podemos obter um limite melhor sobre o valor de $PCV_s$ se incluirmos as restrições (II.15b) na f.o de $P_T$ pelo uso de multiplicadores de Lagrange $\lambda$, ou seja; resolvendo-se o problema:

$$\underset{x}{\text{minimizar}} \left( \sum_{\ell=1}^{m} c_\ell x_\ell \right) + \sum_{i \in N} \lambda_i \left( \sum_{\ell \in A_i} x_\ell - 2 \right)$$

s.a.

$$P_T(\lambda) \begin{cases} \underset{x}{\min} \left[ \sum_{\ell=1}^{m} (c_\ell + \lambda_{i_\ell} + \lambda_{j_\ell}) - 2 \sum_{i \in N} \lambda_i \right] \\ \text{s.a. restrições (II.2)-(II.5)} \end{cases}$$

onde $i_\ell$ e $j_\ell$ são vértices terminais do arco $\ell$.

Temos então que os problemas $P_t(\lambda)$ e $P_t$ são iguais para qualquer $\lambda$ mas $P_t$ tem os custos transformados para $c_\ell = c_\ell + \lambda_{i_\ell} + \lambda_{j_\ell}$.

Os melhores valores das penalidades $\lambda^*$ podem ser obtidos dentre os quais maximizam a seguinte expressão:

$$V(P_t(\lambda^*)) = \max_{\lambda} \left[ V(P_t(\lambda)) \right] \tag{II.16}$$

**Notação**: V(.) será usado para representar o ótimo do problema (.).

Tem-se então que $V(P_t(\lambda^*))$ é um limitante inferior para $V(PCV_s)$.

HELD e KARP (1970) e CHRISTOFIDES (1970) foram os primeiros a proporem este procedimento com o uso de penalidades.

Em geral, é simples provar que existe um "gap" positivo g entre o valor do limite $V(P_t(\lambda^*))$ e a solução ótima do PCV, $V(PCV_s)$. Entretanto, g é freqüentemente menos do que 1% do valor de $V(PCV_s)$. Verifica–se que problemas com mais de 100 cidades têm sido resolvidos satisfatoriamente introduzindo o limite $V(P(\lambda^*))$ nos algoritmos "branch and bound" [CHRISTOFIDES, 1979].

**PCV Assimétrico**

As mesmas suposições do caso PCV simétricas são válidos, apenas nosso grafo G agora é direcionado.

Se um arco (i, v) incidente em v é removido de r, o grafo restante é um caminho direcionado (com raiz no vértice v) passando por todos os vértices de G. Claramente, se $T_V$ é a árvore geradora mínima direcionada com raiz em v, então o custo em $T_V$ mais o custo do arco (i, v) de menor peso – dito $(i_1, v)$ – é um limite inferior sobre o custo de r.

Chamemos de $G_v$ o grafo composto pelos arcos de $T_v$ e o arco $(i_1, v)$. O problema cuja solução ótima é o grafo $G_v$ é:

$$
P_{DT} \quad
\begin{array}{lll}
\text{minimizar} & \displaystyle\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}\, x_{ij} & \\
\text{s.a.} & \displaystyle\sum_{i=1}^{n} x_{ij} = 1 & \forall\, j \in V \\
& \displaystyle\sum_{i \in S_t} \sum_{j \in \bar{S}_t} x_{ij} \geq 1 & \forall\, S_t \subset V \\
& x_{ij} \in \{0,1\} & \forall\, i,j \in V
\end{array}
$$

Seja $PCV_a$ o PCV assimétrico composto das restrições (II.6)–(II.10a). Observa–se facilmente que o problema $P_{DT}$ é uma relaxação do $PCV_a$ onde as restrições (II.8) foram esquecidas.

Novamente, um melhor limite do que $V(P_{DT})$ sobre $V(PCV_a)$ pode ser obtido pela relaxação lagrangeana da restrição (II.3), que produzirá o seguinte problema

$$P_{DT}(\lambda): \begin{cases} \underset{x}{\text{minimizar}} \ \sum_{i=1}^{n} \sum_{j=1}^{n} (c_{ij}+\lambda_i)\, x_{ij} - \sum_{i=1}^{n} \lambda_i \\ \text{s.a.} \\ \text{restrições (II.7), (II.9) e (II.10a)} \end{cases}$$

$$V(P_{DT}(\lambda^*)) = \max_{\lambda} [V(P_{DT}(\lambda))] \tag{II.17}$$

é um limite inferior do $V(PCV_a)$.

CHRISTOFIDES (1979) afirma que para um dado $\lambda$, $P_{DT}(\lambda)$ é o problema de achar a árvore geradora direcionado de custo mínimo com raiz v do grafo G com custos modificados $c_{ij}$. Afirma também que pode ser resolvido facilmente por um algoritmo limitado polinomialmente como mostrado por EDMONDS (1967) e FULKERSON (1974).

No caso assimétrico, desvantajosamente, temos uma dificuldade maior para achar a árvore geradora mínima direcionada do que no caso do grafo não direcionado. O cálculo do limite $V(P_{DT}(\lambda^*))$ da equação (II.17) é muito mais custoso do que o cálculo do limite $V(P_T(\lambda^*))$ da equação (II.16).

A qualidade do limite $V(P_{DT}(\lambda^*))$ é inferior à qualidade do limite $V(P_t(\lambda^*))$

para o problema simétrico.

Para os casos em que o número de cidades do PCV assimétrico excede 60, o limite $V(P_{DT}(\lambda^*))$ não pode ser usado pelas desvantagens expostas acima [SMITH e THOMPSON, 1975].

### II.2.2 – Limites Obtidos do Problema da Designação (PD)

Esses limites são obtidos da seguinte maneira (CHRISTOFIDES, 1979):

O problema da designação definido pelas restrições (II.6)–(II.9). Observa–se que esta é uma relaxação do $PCV_a$ onde as restrições (II.10a), (II.10b) ou (II.10c) foram abandonadas.

Balas e Christofides utilizaram relaxação lagrangeana para algumas restrições violadas em (II.10a) e algumas outras de (II.10c).

Seja $\lambda_t$ o multiplicador associado com a t–ésima restrição em (II.10a) que não é satisfeita. Temos então o problema:

$$P_D(\lambda)\begin{cases} \underset{x}{\text{minimizar}} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}\, x_{ij} - \sum_{t} \lambda_t \sum_{i \in S_t} \sum_{j \in \bar{S}_t} x_{ij} + \sum_{t} \lambda_t \\ \text{s.a.} \\ \text{restrições (II.7)–(II.9)} \end{cases}$$

Esta relaxação apresenta o inconveniente de poder ter o número de multiplicadores $\lambda_t$ na f.o. de $P_D(\lambda)$ aumentado exponencialmente. Portanto, se h for o número de circuitos formados pela solução do PD, então existem $2^{h-1} - 1$ possíveis conjuntos $S_t$ para os quais a restrição (II.10a) pode ser violada; um para o corte

$(S_t, \bar{S}_t)$ e um para o corte $(\bar{S}_t, S_t)$ o que irá produzir um total máximo de $2^h-2$ restrições violadas ao tipo (II.10a). Desde que o número de circuitos (h) pode ser tão grande quanto, $\lfloor n/3 \rfloor$, o número de restrições violadas é exponencial em n.

Para a escolha das restrições a entrar na f.o. com o uso da relaxação lagrangeana, BALAS e CHRISTOFIDES (1976), usaram um procedimento sequencial e desenvolveram um método não iterativo aproximado para determinar os multiplicadores de lagrange como veremos a seguir.

Seja $[\bar{c}_{ij}]$ um elemento da matriz de custos reduzidos encontrada após a solução do PD. Determine o grafo $G_0 = (V, A_o)$ onde $A_0 = \{(i,j)/\bar{c}_{ij} = 0\}$.

Escolha qualquer vértice $i \in V$ e forme o conjunto de vértices R(i) que podem ser alcançados a partir do vértice i através de arcos de $G_0$.

Se o conjunto obtido R(i) for igual a V, é feita a escolha de um outro vértice e a operação acima é repetida. Se $R(i) = V \ \ \forall i \in V$ pare: nenhum outro corte poderá ser gerado. No caso em que $R(i) \neq V$ geramos um corte $K_t = (S_t, \bar{S}_t)$ com $S_t = R(i)$ que viola (II.10a).

Podemos calcular um valor inicial do multiplicador $\lambda_t$, uma vez que já dispomos de $K_t$ como sendo:

$$\lambda_t^0 = \min_{(i,j) \in K_t} [\bar{c}_{ij}]$$

Os outros [i,j] são atualizados por:

$$\bar{c}_{ij} = \bar{c}_{ij} - \lambda_t^0 \quad \forall (i,j) \in K_t$$

$$\bar{c}_{ij} = \bar{c}_{ij} \quad c/c$$

e o conjunto $A_o$ também é atualizado para incluir os novos arcos que tem $\bar{c}_{ij} = 0$.

Repete–se este procedimento até que nenhum corte mais possa ser gerado. O número máximo de cortes que podem ser gerados é 2(h–1).

Agora já não se pode melhorar $V(P_D(\lambda))$ pela modificação de $\lambda$ e ao mesmo tempo mantendo os custos modificados $\bar{c}_{ij}$ não negativos. Temos ainda, que a solução do problema $P_D$ é ainda ótima para o problema $P_D(\lambda)$ e é possível identificar as restrições da forma (II.10c) que são violadas por esta solução.

Vamos supor que a solução contém circuitos $r_p$, onde $r_p$ é o conjunto dos arcos que formam o circuito.

Utilizando a relaxação lagrangeana das correspondentes restrições (II.10c) teremos o problema:

$$P_D(\lambda,\mu) \qquad \begin{aligned} &\underset{x}{\text{minimizar}} \ \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}\, x_{ij} - \sum_t \lambda_t \sum_{i\in S_t}\sum_{j\in \bar{S}_t} c_{ij} \\ &\qquad + \sum_p \mu_p \sum_{(i,j)\in r_p} x_{ij} - \sum_p \mu_p\, |r_p| + \sum_t \lambda_t + \sum_p \mu_p \\ &\text{s.a.} \\ &\text{restrições (II.7)–(II.9)} \end{aligned}$$

Se assumirmos que $\lambda$ estão fixados em $\lambda_t$, como mencionado anteriormente, podemos considerar de modo similar ao usado para calcular os $\lambda$ do problema de maximizar $V(P(\lambda,u)\ \forall\mu$, ou seja, atribuindo o valor fixo $\mu_p^0$ para $\mu_p$ que é o maior valor de $u_p$ tal que a solução do PD para o problema $P(\lambda,u)$ não é modificada. Podemos calcular o valor de $\mu_p^0$ da mesma maneira simples como os duais são calculados para o Problema da Designação.

Já dispomos de $\mu_p^0$, agora calcularemos as novas variáveis duais $u_i$ e $v_j$ para o problema $P_D(\lambda^0, \mu_p^0)$. Podemos usar as variáveis duais para atualizar os custos $\bar{c}_{ij}$ correspondentes ao problema $P_D(\lambda^0)$ por $\bar{c}_{ij} = \bar{c}_{ij} - u_i - v_j$.

Identificaremos outro circuito $r_p$ na solução do PD e determinaremos outro valor de $\mu_p^0$. Calcularemos novas variáveis $u_i$ e $v_j$ para o problema $P_D(\lambda^0,[\mu_{p_1}^0, \mu_{p_2}^0)]$ como $p_1$ e $p_2$ sendo os índices do primeiro e do segundo circuitos considerados; atualizaremos também os custos $\bar{c}_{ij}$.

Este procedimento é repetido até que todos os circuitos $r_p$ que formam a solução do PD tenham sido considerados.

No final do procedimento, uma vez que temos que a solução inicial do PD ainda é ótima, o valor

$$V(P_D) + \sum_t \lambda_t^0 + \sum_p \mu_p^0$$

é um limitante inferior para o PCV e é uma aproximação razoável para o valor de:

$$\max_{\lambda, \mu} [V(P_D(\lambda, \mu))]$$

Suponhamos que B seja uma solução ótima para o problema da designação $P_D(\lambda^0, \mu^0)$. Como já sabemos que a solução inicial do PD é ótima, o que estamos procurando são outras soluções alternativas. Se B é um circuito hamiltoniano que satisfaz as restrições para as quais $\lambda \neq 0$ e $\mu \neq 0$ são igualdades, então B é uma solução ótima para o PCV.

Se B é um circuito hamiltoniano mas algumas restrições com $\lambda$, $\mu \neq 0$ não são satisfeitas quando igualdades, então poderia ainda ser possível modificar $\lambda$ e $\mu$ e obter um limite inferior melhor do que o valor da melhor solução conhecida na árvore de busca. BALAS e CHRISTOFIDES (1976) desenvolveram algumas técnicas para modificar $\lambda$ e $\mu$ para melhorar o limitante inferior.

No caso de B não ser um circuito hamiltoniano, faremos uso novamente do grafo $G_0 = (X, A_0)$, $A_0 = \{(i,j)/\bar{c}_{ij} = 0\}$, onde $\bar{c}_{ij}$ são os custos reduzidos do problema $P_D(\lambda^0, \mu^0)$.

Seja $\hat{r}$ a solução ótima para o PCV. $\hat{r}$ define então um grafo $G(\hat{r})$, consistindo de um único circuito hamiltoniano.

Se removermos um vértice v do grafo $G(\hat{r})$ o grafo resultante $G_v(\hat{r})$ possui um caminho que passa por todos os $(n-1)$ e vértices restantes. Este é um grafo conectado unilateralmente, o que quer dizer que para quaisquer dois vértices i e j de $G_v(\hat{r})$ deve existir um caminho de i para j ou de j para i.

Consideremos o grafo $G_0^V = (X^V, A_0^V)$, onde o vértice v foi removido. Se $G_0^V$ não é unilateralmente conectado então deve existir um par de cortes $K_V^{\prime}$, $K_V^{\prime\prime}$ de $G_0^V$ para os quais:

$$A_0^V \cap K_V^{\prime} = A_0^V \cap K_V^{\prime\prime} = \phi$$

Seja

$$\pi_v = \min_{(i,j) \in K'_V \cup K''_V} [\bar{c}_{ij}]$$

Os novos custos reuzidos serão dados por

$$\bar{c}_{ij} = \begin{cases} \bar{c}_{ij} - \pi_v, & \forall (i,j) \in K'_V \cup K''_V \\ \bar{c}_{ij}, & \text{caso contrário} \end{cases}$$

Podemos definir um novo grafo $G_0$ que poderá ser testado em relação à conectividade unilateral após a remoção de algum outro vértice V. Continuaremos o processo até que $G_0^V$ continue conectado unilateralmente depois da remoção de qualquer V.

Seja $W = \{v_1, v_2, \ldots, v_k\}$ o conjunto de vértices cuja remoção deixa $G_0^V$ não conectado unilateralmente durante este procedimento. Então a quantidade

$$B_{PD} = V(P_D) + \sum_t \lambda_t^0 + \sum_p \mu_p^0 + \sum_{v \in \omega} \pi_v$$

é um limitante inferior para o PCV.

Com relação aos limites obtidos do problema da designação podemos afirmar:

i) estes limites podem ser obtidos de problemas simétricos e assimétricos embora com melhores resultados no último caso;

ii) o valor obtido para problemas simétricos do problema da designação $(V(P_D))$

é, em média 20% abaixo da solução ótima do PCV simétrico ($V(PCV_s)$) enquanto $B_{PD}$ é, em média, apenas 4% abaixo de $V(PCV_s)$. Temos ainda que o limite $B_{PD}$ é muito inferior ao limite obtido da árvore mínima geradora ($V(P_t(\lambda^*))$) embora seja 4 a 5 vezes mais rápido calcular $B_{PD}$ do que $V(P_t(\lambda^*))$ para problemas simétricos na faixa até 100 cidades;

iii) no caso de problemas assimétricos, o valor da solução do problema da designação $V(P_D)$ é, em média, cerca de 3% abaixo do valor ótimo do PCV assimétrico ($V(PCV_a)$), $B_{PD}$ é, média, apenas cerca de 0,5% abaixo de $V(PCV_a)$. Temos ainda no caso de problemas assimétricos que o limite $B_{PD}$ é da mesma qualidade do limite obtido na árvore geradora mínima direcionada $V(P_{DT}(\lambda^*))$ apresentando ainda a vantagem de requerer em média 10 a 20 vezes menos esforços computacional.

iv) no caso de problemas assimétricos com mais de 250 cidades Balas e Christofides os resolveram utilizando limites baseados no PD em algoritmos de busca em árvore com restrições de eliminação de subrotas.

### II.2.3 – Limites Obtidos do Problema de Matching (PM)

Consideremos o seguinte problema 2–matching:

$$\min \sum_{\ell=1}^{m} c_\ell x_\ell \qquad \text{(II.18)}$$

$$P_M \quad \text{s.a.} \sum_{\ell \in A_i} x_\ell = 2 \qquad \text{(II.19)}$$

$$x_\ell \in \{0,1\} \qquad \text{(II.20)}$$

Observa–se que o problema formulado acima é o $PCV_s$ que foi definido por (II.11), (II.15a) e (II.15b) tendo a restrição (II.15a) sido ignorada. Se incluirmos (II.15a) na função objetivo com uso de um multiplicador de lagrange teremos:

$$P_M(\lambda)\begin{cases} \min\limits_{x} \sum\limits_{\ell=1}^{m} c_\ell x_\ell - \sum\limits_{t} \lambda_t \sum\limits_{\ell\in k_t} x_\ell + \sum\limits_{t} \lambda_t \\ \text{s.a.} \\ \text{restrições (II.19) e (II.20)} \end{cases}$$

Como este problema tem essencialmente a mesma forma do problema $P_D(\lambda)$ sugerimos que o leitor utilize as mesmas técnicas que foram usadas para calcular o limite $B_{PD}$ para obter um limite análogo $B_M$.

O limite obtido do problema de matching é de qualidade superior ao obtido pelo Problema da Designação para PCV simétricos.

## II.3 – REGRAS DE BRANCHING

Apesar de não ter uma importância tão grande em relação à eficiência dos algoritmos como os limitantes obtidos através das diversas relaxações, podemos afirmar que a escolha de uma boa regra de "branching" nos proporcionará um menor número de subproblemas a resolver. Portanto uma boa regra de "branching" deve ter dois aspectos principais; como sugerido por BALAS e TOTH (1985):

a) gerar poucos sucessores de um nó na árvore de busca;

b) gerar subproblemas fortemente restritos, isto é, excluir muitas soluções de cada subproblema.

Apesar destes aspectos serem conflitantes na maioria das vezes os méritos das diversas regras residem em conciliar estes fatores.

Apresentaremos a seguir algumas regras de "branching" que estão separadas de acordo com as relaxações com que estão relacionadas.

As regras de separação que veremos têm em comum os conjuntos de arcos $E_k$ e $I_k$ dos arcos excluídos ou incluídos na solução do subproblema K respectivamente, que foram reunidas por BALAS e TOTH (1985).

### II.3.1–Regras de Separação Relacionadas com o Problema da Designação (Relaxação)

**Regra 1:** [LITTLE, MURTY, SWEENEY & KAREL, 1963]

Dado o subproblema relaxado corrente $PD_k$ e seus custos reduzidos $\bar{c}_{ij} = c_{ij} - u_i - v_j$ onde $u_i$ e $v_j$ são variáveis duais ótimas, para cada arco (i,j) tal que $\bar{c}_{ij} = 0$ defina a penalidade

$$p_{ij} = \min\{\bar{c}_{ih} \mid h \in V - \{j\}\} + \min\{\bar{c}_{hj} \mid h \in V - \{i\}\}$$

e escolha um arco $(r, s) \in A - E_k U_k$ tal que:

$$p_{rs} = \max\{p_{ij} \mid \bar{c}_{ij} = 0\}$$

Dessa maneira geramos dois sucessores do nó K que são os nós K1 e K2 definindo:

$$E_{K1} = E_K \cup \{(r,s)\}, \; I_{K1} = I_K$$

e

$$E_{K2} = E_{K'} \quad I_{K2} = I_{K2} \cup \{(r,s)\}$$

Esta regra pode ser aplicada a qualquer problema inteiro pois não leva em consideração a estrutura especial do PCV e ainda tem a desvantagem de permitir que a solução ótima do $PD_K$ seja viável para $PD_{K2}$.

**Regra 2:** [BELLMORE & MALONE, 1971]

Seja $x^k$ a solução ótima do subproblema relaxado corrente $PD_K$ e seja $A_S = \{(i_1, i_2), ..., (I_t, i_1)\}$ o conjunto de arcos de uma subrota de cardinalidade mínima de $x^k$, envolvendo o conjunto de vértices $S = \{i_1, ..., i_t\}$. A restrição (II.10a) implica na disjunção

$$(x_{i_1 j}=0,\ j \in S) \vee (x_{i_2 j}=0,\ j \in S) \vee \ldots \vee (x_{i_t j}=0,\ j \in S) \qquad \text{(II.21)}$$

Gera t sucessores do nó K, definido por

$$\left.\begin{array}{l} E_{kr} = E_k \cup \{(i_r, j) \mid j \in S\} \\ I_{kr} = I_k \end{array}\right\} r = 1, ..., t \qquad \text{(II.22)}$$

Esta regra torna $x^k$ inviável para todos os problemas sucessores de $PD_k$ mas (II.21) não particiona o conjunto viável de $PD_k$. Poderemos remediar isto na regra seguinte, que difere apenas no modo como define a partição:

**Regra 3:** [GARFINKEL, 1973]

A disjunção (II.21) pode ser fortalecida para:

$(x_{i_1j} = 0,\ j \in S) \vee (x_{i_1j} = 0,\ j \in V-S;\ x_{i_2j} = 0,\ j \in S) \vee$

$\vee\ (x_{i_rj} = 0,\ j \in V-S,\ r = 1, \ldots, t-1;\ x_{i_tj} = 0,\ j \in S)$

Tendo então:

$E_{kr} = E_k \cup \{(i_r,j) \mid j \in S\} \cup \{(i_q,j) \mid q = 1, \ldots, r-1,\ j \in V-S\}$

$I_{kr} = I_k$

Esta regra exclui um grande número de designações do conjunto de soluções viáveis de cada problema sucessor.

## II.3.2 – Regras de Separação Relacionadas com o Problema da 1–Árvore

**Regra 4:** [HELD & KARP, 1971]

No nó k, sejam as arestas livres da 1–árvore corrente (ou seja, são aquelas em $A - E_k \cup I_k$) ordenadas de acordo com as penalidades não–crescentes e sejam os primeiros q elementos deste conjunto ordenado reunidos no conjunto $J = \{\{c_1, j_1\}, \ldots, \{i_q, j_q\}\}$ onde q será especificado abaixo. Defina q novos nós $k_1, \ldots, k_q$ por

$I_{kr} = I_k \cup \{\{i_h, j_h\} \mid h = 1, \ldots, r-1\},\quad r = 1, \ldots, q$

$$E_{kr} = E_k \cup \{\{i_r, j_r\}\}, \quad r = 1, ..., q-1$$

$$E_{kq} = E_k \cup \{\{i, j\} \notin I_{kq} | i = p \text{ ou } j = p\}$$

Aqui $p \in V$ é tal que $I_k$ contém no máximo uma aresta incidente com p, enquanto $I_{kq}$ contém duas destas arestas; e q é o menor sub–índice de uma aresta em J para qual um vértice com as propriedades de p existe.

**Regra 5:** [SMITH & THOMPSON, 1977]

Escolha um vértice i de G cujo grau na 1–árvore corrente não seja 2 e uma aresta de custo máximo {i, j} na 1–árvore. Gere então subproblemas definidos por

$$E_{k1} = E_r \cup \{\{i, j\}\}, \quad I_{k1} = I_k$$

$$E_{k2} = E_k, \quad I_{k2} = I_k \cup \{\{i, j\}\}$$

Esta regra gera apenas dois sucessores de cada nó na árvore de busca, mas a 1–árvore mínima no problema k continua viável para o problema $K_2$.

## II.4 – ENUMERAÇÃO IMPLÍCITA

Como dissemos anteriormente, os algoritmos "branch and bound" que foram desenvolvidos para Problemas Lineares Inteiros binários, devido às suas particularidades são distinguidos do método "branch and bound".

**Algoritmo de Balas**

Podemos catalogá–lo como de busca em árvore, onde cada iteração pode ser vista como ramificação em árvore binária.

Seja o seguinte problema:

O problema pode ser definido por:

$$\text{minimizar } z = \sum_{j=1}^{n} c_j\, n_j \qquad \text{(II.21)}$$

P: s.a.

$$\sum_{j=1}^{n} a_{ij}\, x_j \leq b_i \quad i = 1, \ldots, m \qquad \text{(II.22)}$$

$$x_j \in \{0,1\} \quad j = 1, \ldots, m \qquad \text{(II.23)}$$

onde $c_j$, $a_j$, $b_i \in \mathbb{R}$.

Podemos supor, sem perda de generalidade, que $c_j \geq 0$, $j = 1, \ldots, n$ pois sempre poderemos substituir a variável $x_k$ que tem $c_k < 0$ por $(1 - x_k^{\prime})$, onde $x_k^{\prime} \in \{0,1\}$.

**Notação:**

Uma solução de (II.23) será representado por

$$x^p = (x_1^p, x_2^p, \ldots, x_n^p)^T \text{ ou}$$

$J_p = \{j | x_j = 1\}$

**Definição**: Uma solução $x^p$ é descendente de $x^q$ se

$J_q \subset J_p$

**Exemplo**

$x^q = (0\ \ 0\ \ 1\ \ 0\ \ 1\ \ 1)^T, \quad x^p = (1\ \ 1\ \ 1\ \ 0\ \ 1\ \ 1)^T$

$x^p$ é descendente de $x^q$

**Propriedade 1**

Se $x^p$ é uma solução de (II.23) então

$$\sum_{j \in J_p} c_j\, x_j \leq \sum_{j \in J_k} c_j\, x_j$$

para todas as soluções $x^k$ descendentes de $x^p$.

**Propriedade 2**

Se $x^0 = (0\ \ 0\ \ \ldots\ \ 0)^T$ satisfaz (II.22) então

$x^0$ é uma solução ótima.

Faremos agora a suposição que a solução inicial $x^0$ seja tal que $J_0 = \phi$, isto é

$x^0 = (0\ 0\ \ldots.\ 0)^T$ e o valor ótimo $V(P) = 0$, (ou $z^* = 0$). Se esta solução for viável, pela Proposição 2, ela é ótima. Caso contrário, se nenhuma solução inicial é óbvia teremos $J_0 = \phi$ e $z^* = \infty$.

**Idéia Geral do Algoritmo de Balas**

À medida que a enumeração é feita, a solução viável tendo melhor valor para a f.o. é armazenada. Desde que a enumeração termina quando todos os $2^n$ possíveis pontos solução foram considerados (implícita ou explicitamente) a última solução viável armazenada (se existir) é a solução ótima. Se nenhuma existir, o problema não tem solução viável.

**Esquema de Enumeração**

Supor que estamos na solução $x^p$ de (II.25) e que $z^*$ seja a melhor solução viável até o momento, isto é, existe $x^k$ tal que $z^* = \sum_{j \in J_k} c_j$ e que $x^k$ seja viável para P. Caso não tenhamos encontrado uma solução viável então $z^* = +\infty$.

A partir da solução $x^p$ desejamos obter $x^q$ descendente de $x^p$ tal que $|J_q| = |J_p| + 1$, ou seja $J_q = J_p \cup \{\ell\}$, $\ell \notin J_p$.

A eficiência do algoritmo depende diretamente da escolha da variável a entrar na base, pois esta escolha permitirá que se avance mais ou menos na árvore de busca.

**Condições de Parada**

i) se $x^p$ é viável de P, não nos interessa buscar uma descendente $x^q$ de $x^p$, pois pela proposição 1 temos $\sum_{j \in J_p} c_j \leq \sum_{j \in J_q} c_j$;

ii) se $\sum_{j \in J_p} c_j + c_\ell \geq z^*$ $\forall \ell \in J_p$, isto quer dizer que todas as soluções descendentes de $x^p$ fornecerão valores da f.o. sempre superiores ou iguais a $z^*$, não havendo interesse em enumerar descendente de $x^p$;

iii) se existir um i tal que:

$$b_i - \sum_{j \in J_p} a_{ij} - \sum_{j \notin J_p} \min\{0, a_{ij}\} < 0$$

não haverá descendente de $x^p$ viável, neste caso não há interesse em enumerar descendente de $x^p$.

i, ii e iii são condições de parada na solução de $x^p$.

Todas as soluções descendentes de $x^p$ estarão **enumeradas implicitamente** quando uma das três condições de parada for satisfeita em $x^p$.

Se nenhuma condição de parada for satisfeita, devemos procurar uma solução descendente de $x^p$, por exemplo $x^q$ tal que $J_q = J_p \cup \{\ell\}$.

Consideremos $s_i = b_i - \sum_{j=1}^{n} a_{ij} x_j$ e $s_i \geq 0$, $i = 1, \ldots, m$

$s_i \rightarrow$ variável de folga da restrição i

Seja $s_i^p = b_i - \sum_{j \in J_p} a_{ij}$, i.e., $s_i^p$ representa o valor de $s_i$ quando $x = x^p$.

Em $x^p$ podemos definir os seguintes conjuntos:

$$A_p = \{k / \sum_{j \in J_p} c_j + c_k \geq z^* \; k \notin J_p\}$$

$$D_p = \{k / \text{se } \forall i \text{ com } s_i^p < 0, \; a_{ik} \geq 0, \; k \notin J_p\}$$

$$C_p = \{1, 2, ..., n\} - (J_p \cup A_p \cup D_p)$$

Utilizando o conjunto $C_p$, poderemos fazer a escolha do índice da variável candidata a tomar valor 1, isto é $\ell \in C_p$ para formar $J_q = J_p \cup (\ell)$.

Seja $d_j^p = \sum_{i=1}^{m} \min\{0, s_i^p - a_{ij}\}$, $j \in C_p$ e $d_\ell^p = \max_{j \in C_p} \{d_j^p\}$.

Cado $d_\ell^p = 0$ estão a solução descendente associada a $J_q = J_p \cup \{\ell\}$ será viável. Se houver mais de um índice para o qual $d_j^p = 0$, i.e., $L_p = \{j \in C_p / d_j^p = 0\}$, então o $\ell$ escolhido para formar a solução descendente será o $\ell$ associado a

$$c_\ell = \min_{j \in L_p} \{c_j\}$$

Se nenhuma condição de parada for verificada, $C_p \neq \phi$. Suponhamos agora que em $x^q$ descendente direto de $x^p$ o conjunto $C_q = \phi$ ou uma das três condições de para seja verificada. Teremos então que voltar de $x^q$ para $x^p$ e atualizaremos $C_p$ de duas maneiras:

i) $C_p := C_p - \{\ell\}$ onde $\ell$ é tal que $J_q = J_p \cup \{\ell\}$

ii) $A_p$ será modificado caso $z^*$ também o seja, acarretando outra modificação em $C_p$.

O retorno de $x^q$ a $x^p$ é denominado "backtracking".

**Estrutura de pilha proposta por Glover e Geoffrion**

Esta pilha representa o conjunto dos índices associados às variáveis fixadas.

Seja a pilha $\pi$, para a qual p(j) será sua j–ésima componente tal que:

$p(j) > 0 \quad \text{se } x_{p(j)} = 1$

e

$p(j) < 0 \quad \text{se } x_{p(j)} = 0$

Por exemplo, $\pi = [-3, 2, -7, -4]$

$x_3 = 0$, $x_2 = 1$, $x_7 = 0$ e $x_4 = 0$ são os valores fixados, todos os descendentes $x^k$ dessa solução não poderão ter 3, 7 e 4 pertencendo a $J_k$.

**Algoritmo de Balas**

**Fase 0 (Inicialização)**

$\pi = \phi;\ z^* = +\infty$

**Fase 1**

Se uma das condições de parada for verificada (no caso de ser a primeira, i.e., $\pi$ está associado a uma solução viável de P, $x^P$, neste caso, se $cx^P < z^\#$, atualizamos $z^\#$ como sendo $z^\# = cx^P$ tendo como melhor solução até o momento $x^P$); ir para fase 2; caso contrário ir para fase 3.

**Fase 2 ("back tracking")**

Enquanto o último elemento da pilha for negativo, removê—lo da pilha. Se a pilha for vazia, ir para fase 4. Caso o último elemento da pilha for positivo, trocar seu sinal e ir para fase 1.

**Fase 3**

Escolher uma variável $x_k$ pelo critério aconselhado e junte à pilha o elemento k, ir para fase 1.

**Fase 4**

Pare, solução ótima associada a $z^\#$; se $z^\# = +\infty$, o problema P não admite solução viável.

# CAPÍTULO III

# MÉTODOS APROXIMADOS

## III.1 – INTRODUÇÃO

Devido à complexidade dos problemas de Otimização Combinatória, dentre os quais se inclue o PCV, uma estratégia que vem sido muito utilizada por diversos pesquisadores é desenvolver algoritmos que produzem soluções boas (quase ótimas) com esforço computacional muito inferior ao que seria necessário se optassem só por soluções ótimas. Estes algoritmos são denominados aproximados.

Na literatura estes algoritmos são agrupados em duas grandes categorias, que passaremos a apresentar aqui, embora ainda possamos destacar os algoritmos de duas fases e os de otimização incompleta. Referências aos dois últimos tipos de algoritmos são encontradas em GILLETT & MILLER (1974).

i) **Algoritmos construtivos**

As rotas vão sendo formadas à medida que o algoritmo vai sendo executado.

ii) **Algoritmos de Melhoria Iterativa**

À partir de uma solução viável inicial o algoritmo tenta melhorá–lha através de algumas modificações locais.

## III.2 – ALGORITMOS CONSTRUTIVOS

Os algoritmos construtivos mais destacados são os de inserção e os de economia.

Um algoritmo de inserção bastante utilizado é o chamado "vizinho mais próximo". Os algoritmos construtivos geram uma rota ótima à partir da matriz de distâncias.

Assumiremos na apresentação destes algoritmos que os custos são simétricos ($c_{ij} = c_{ji}$) e satisfazem a desigualdade triangular e são definidos para cada aresta (i, j) à menos que especifiquemos o contrário.

### III.2.1 – Regra do Vizinho mais Próximo (VMP)

Começa–se com um vértice sendo escolhido arbitrariamente e prossegue–se para formar um caminho onde o próximo vértice a ser adicionado é o vizinho mais próximo, ou seja, é o de menor custo, que ainda não foi adicionado até que todos os vértices sejam usados. Os dois vértices extremos do caminho são então ligados para obtermos a solução do PCV.

Este é um algoritmo do tipo construtivo, onde a solução vai sendo obtida passo a passo pela inserção na solução ótima do vértice mais próximo do último vértice inserido. Como as rotas vão sendo obtidas na medida da execução do algoritmo, uma interrupção precoce do método não irá fornecer uma solução viável.

ROSENKRANTZ, STEARNS e LEWIS (1974) mostraram que:

$$\frac{V(VMP)}{V(PCV)} \leq \frac{1}{2}\left(\lceil \log n \rceil + \frac{1}{2}\right)$$

onde V(VMP) é o valor obtido pela regra do vizinho mais próximo V(PCV) é o valor da solução ótima do PCV e $\lceil x \rceil$ é o menor inteiro maior ou igual do que x e n é o número de nós.

Criaram o seguinte algoritmo que apresentaremos simplicadamente:

Procedimento vizinho mais próximo

**Passo 1:** Começa com algum nó sendo escolhido como o início ao caminho.

**Passo 2:** Acha o nó mais perto do último nó adicionado ao caminho. Acrescenta este nó ao caminho.

**Passo 3:** Repete o passo 2 até que todos os nós estão contidos no caminho. Liga então o primeiro ao último nó.

De um ponto de vista computacional o procedimento ainda pode ser repetido n vezes cada vez com um nó sendo escolhido como o nó inicial. A melhor solução obtida pode então ser listada como a resposta. Utilizando–se esta estratégia o tempo gasto será proporcional a $n^3$.

No caso em que $n \geq 15$, eles também mostraram que

$$\frac{V(VMP)}{V(PCV)} > \frac{1}{3}\left(\log(n+1) + 4/3\right) \tag{III.1}$$

e observaram que a causa do limite do pior caso sobre V(VMP) dado por (III.1) aumentar logaritmicamente com n não é devido ao último arco adicionado no VMP ser muito grande nem pela escolha arbitrária do vértice inicial.

Para executarmos este algoritmo são necessárias da ordem de $0(n^2)$ operações.

### III.2.2 – Algoritmos de Inserção

Um algoritmo de inserção escolhe uma subrota de k nós na iteração k e tenta determinar que nó (que ainda não está na subrota) pode ser o próximo na rota (passo de seleção) e então determina onde ele poderia ser colocada na subrota (passo de inserção).

#### III.2.2.1 – Inserção do Mais Próximo (IP)

**Passo 1:** Começa com um subgrafo consistindo do nó i apenas.

**Passo 2:** Acha um nó k tal que $c_{ik}$ seja mínimo e forma a subrota i–k–i.

**Passo 3:** (Passo de seleção)

Dada a subrota, ache o nó k que não está na subrota mais próxima de algum nó na subrota.

**Passo 4:** (Passo de inserção)

Acha o arco (i,j) na subrota que minimiza $c_{ik} + c_{kj} - c_{ij}$.

Insere k entre i e j.

**Passo 5:** Vá para o passo 3 a menos que tenhamos um ciclo hamiltoniano.

Pior caso:

$$\frac{V(IP)}{V(PCV)} \leq 2$$

Este procedimento requer da ordem de $n^2$ cálculos.

### III.2.2.2 – Inserção dos Mais Barato

Tem um procedimento semelhante ao anterior exceto nos passos 3 e 4 que são substituídos por:

**Passo 3':** Acha (i,j) na subrota e um k não pertencente tal que $c_{ik} + c_{kj} - c_{ij}$ seja mínimo e então insira k entre i e j.

Apresenta um comportamento no pior caso igual ao anterior e requer da ordem de $n^2 \log(n)$ cálculos.

### III.2.3 – Algoritmos de Economia

O conceito de "economia" (savings) foi introduzido por CLARK & WRIGHT (1964), dando origem à diversos métodos que se basearam nesta idéia.

Apresentaremos resumidamente este procedimento.

**Passo 1:** Seleciona qualquer nó como um depósito central que será denotado como nó 1.

**Passo 2:** Calcula as economias (savings) $s_{ij} = c_{1i} + c_{1j} - c_{ij}$ para $i, j = 2, 3, ..., n$.

**Passo 3:** Ordena decrescentemente as economias numa lista.

**Passo 4:** Começando do topo da lista, faça:

i) se a economia $s_{ij}$ atual fornece uma ligação viável então realiza-se. Caso contrário, rejeite-a;

ii) pegue o próximo da lista e repita o passo 4 i.

Se nenhuma ligação da lista pode ser realizada, PARE: a configuração atual é a solução do problema.

Entende-se por ligação viável uma que fornece uma rota viável de acordo com as restrições do PCV.

GOLDEN (1977) demonstrou que para uma versão sequencial deste algoritmo onde à cada passo selecionamos as melhores economias através do último nó adicionado à subrota a razão de pior caso é limitada por uma função linear em log(n). ONG (1981) demonstrou um resultado similar para a versão paralela.

Número de Cálculos:

O cálculo da matriz $S = [s_{ij}]$ no passo 2 requer área de $cn^2$ operações para alguma constante i. No passo 3 as economias podem ser ordenadas decrescentemente usando o método "Heapsrot"de WILLIAMS (1974) e FLOYD (1964) em um máximo de $cn^2 \log(n)$ comparações e deslocamentos. Passo 4 envolve no máximo $n^2$ operações desde que haja muitas economias a considerar. Logo o procedimento de economias de CLARK & WRIGHT requer da ordem de log(n) cálculos.

## III.3 – ALGORITMOS DE MELHORIA ITERATIVA

### III.3.1 – Algoritmo K–Otimo (LIN, 1965)

Este algoritmo se inicia à partir de uma rota do PCV viável r. Removeremos k arcos de r criando–se então k caminhos desconexos, permitindo–se que alguns desses caminhos possam ser constituídos por um nó isolado.

Ligaremos novamente estes k caminhos de modo a termos uma outra rota viável r', é claro que com arcos diferentes daqueles removidos. Esta operação é chamada uma k–troca. A rota r será descartada se o custo da rota r' for menor. O procedimento será repetido agora para r'. Se o custo de r' não for menor devemos escolher um outro conjunto de k arcos de r para realizar a k–troca.

O procedimento é realizado até que não se possa obter melhorias com a k–troca. Dizemos então que uma rota é k–ótima se nenhuma k–troca produz uma rota de custo menor.

Observa–se facilmente que quanto maior k, melhor será a solução e também maior será o esforço computacional.

As heurísticas 2–ótima e 3–ótima são as mais usadas, como podemos ver na literatura pois ao fazermos a análise de "performance" destas o esforço computacional realizado pela 4–ótima não justifica a melhoria obtida. Já a 3–ótima apresenta resultados bastante superiores aos da 2–ótimas [LIN e KERNIGHAM, 1973].

ROSENKRANTZ, STEARNS e LEWIS (1974) mostraram que para todo $n \geq 8$ existe um grafo para o qual

$$\frac{V(k\text{–ótimo})}{V(PCV)} = 2(1 - \frac{1}{n})$$

para todo $k \leq n/4$.

Apesar de podermos achar uma rota k–ótima com um número de operações polinomial em n, este número é exponencial em k e é limitado por $n^k$. Justifica se assim também porque usamos valores pequenos de k (2 e 3) nas heurísticas [LIN e KERNIGHAN, 1973].

**III.3.2 – Heurística de Christofides (Ch)**

Seja $T^*$ a solução da árvore geradora mínima do grafo G. Relativo a $T^*$, seja $X^0$ o conjunto de vértices com grau impar. Resolva o problema de matching (1–matching) para o grafo $\langle X^0 \rangle$ e seja M o conjunto de arcos neste matching. O grafo $G' = (X, A')$ composto pelo conjunto X de vértices de G e tendo como arcos o conjunto A' formado apenas por aqueles arcos em $T^*$ e M seja euleriano, i.e., todos os vértices são de grau par, podendo portanto, ser atravessado por um circuito euleriano E de modo que cada arco de G' é atravessado uma única vez somente por E. Pode–se construir um ciclo hamiltoniano r de G que serve como uma solução heurística para o PCV usando o circuito E como segue:

Começa construindo r com os seguintes arcos de E. Se um vértice já visitado por r reaparece na seqüência de vértices E, pule este vértice a menos que todos os vértices estajam em r. Neste caso, retornamos ao vértice inicial.

Christofides mostra que:

$$\frac{V(ch)}{V(PCV)} < \frac{3}{2}$$

A heurística necessita da ordem de $0(n^3)$ operações para ser executada.

As heurísticas apresentadas anteriormente são apenas algumas das mais destacadas. Encontra–se na literatura diversas outras que podem ser também aplicadas a resolver os problemas apresentados com igual desempenho.

# CAPÍTULO IV

# SISTEMA DE MANUFATURA FLEXÍVEL

## IV.1 – INTRODUÇÃO

O desenvolvimento de uma nova filosofia para a manufatura de produtos atinge seu ápice com os conceitos que vão ser expostos do Sistema de Manufatura Flexível (SMF).

HARTLEY (1984) afirma que este conceito representa uma nova revolução industrial da mesma importância e impacto que o surgimento da linha de montagem (assembly line) no início do século. Esta filosofia e não apenas a robótica irá revolucionar os processos de manufatura.

Esta nova filosofia de manufatura de produto pode ser definida, segundo Finke, como sendo um sistema integrado de componentes (ferramentas/máquinas), ligados por um sistema de manipulação de materiais e sendo controlados por um sistema computadorizado.

AGOSTINHO (1985) afirma que estas mudanças no conceito de produção estão sendo aplicadas principalmente nos países onde há um grande parque industrial instalado e com mercados extremamente competitivos como é o caso do Japão, Estados Unidos e alguns países europeus. As mudanças podem ser observadas através das seguintes afirmações:

i) diversificação crescente de produtos atendendo exigências do mercado consumidor ao mesmo tempo que diminui a vida dos produtos (novos produtos são lançados em prazos cada vez menores);

ii) procura por aumento de produtividade.

Estas transformações tornaram inadequadas as filosofias de grandes lotes e produção em massa que são características marcantes do estilo de produção americano [AGOSTINHO, 1985].

Observando às novas tendências do mercado surgem novos conceitos de produção na manufatura. Estes conceitos de produção foram desenvolvidos principalmente no Japão e alguns países europeus e compreendem principalmente: sistema de produção em células de manufatura, uso de máquinas de controle numérico e novas técnicas de produção como estoque zero (zero inventory) e produzir somente o necessário (just–in–Time production).

## IV.2 – TIPOS DE SMF E ALGUMAS DEFINIÇÕES

COCA–BALTA (1991) reuniu e traduziu do inglês a maioria dos termos que usamos nesta dissertação sobre o SMF e que passamos a considerar para familiarizar todos os leitores com os novos conceitos que serão apresentados.

Em relação aos processos de produção discreta e levando em consideração a quantidade produzida temos:

**Produção em massa** – caracterizada pela grande quantidade de peças produzidas mas com uma variedade pequena em relação aos tipos. Os produtos produzidos sofrem mudanças apenas à médio ou longo prazo, sendo inviáveis transformações bruscas (curto prazo) no esquema de produção.

**Produção em lotes** – caracterizada pela produção de uma quantidade mediana de peças e também pela média diversificação dos modelos de peças produzidos.

Geralmente, cada operação é executada para todo o lote antes da execução de outra operação.

Produção tipo "job–shop" – caracterizada pela quantidade muito pequena de peças produzidas e pela grande diversificação dos tipos.

Em relação às máquinas usadas.

**máquinas de controle numérico:** se é usado no processo de controle da máquina um programa pré–estabelecido com dados numéricos dizemos que esta máquina é uma máquina de controle numérico. Permite–se então que as operações da máquina sejam realizadas com um mínimo de ações de um operador.

Se um computador for usado para o sistema de controle esta é dita ser uma **máquina de controle numérico computadorizado** (CNC).

**Magazine:** tambor ou cilindro usado para colocar as ferramentas em condições operacionais.

**Tool Slot:** (posição das ferramentas) orifícios ou posições que o magazine possui para montar as ferramentas.

**Fixture:** (fixador) ferramenta de fixação para segurar as peças no processo de usinagem.

**Gripper**: tipo de fixador usado para as peças que necessitam de movimento de rotação para sua usinagem.

**Pallet**: (bandeja) mesa de trabalho ou bandeja onde são colocadas as peças a serem processadas.

**Layout**: desenho ou configuração física do sistema.

**Setup**: tempo gasto para que se possa desenvolver tarefas, de montagem (loading) e desmontagem (unloading) das ferramentas nas máquinas, limpeza do local de trabalho.

**Buffer**: (armazém) local onde são guardadas as peças a serem processadas ou as que já o foram.

**Lead Time**: é o tempo de processamento gasto para fabricação de uma dada peça.

KUSIAK (1986) dividiu em cinco classes os tipos de SMF levando em consideração o número de máquinas de controle numérico e sua disposição (layout):

1) **Módulo de Manufatura Flexível (MMF)**

É composto de uma máquina de controle numérico (CN), um armazém (buffer) de peças, tambor de ferramentas e bandeja (pallets) para o deslocamento de material;

2) **Células (CMF)**

É composta por vários módulos e pode ser construída de acordo com o produto ou tipo de processo.

3) **Grupo (GMF)**

É constituído por várias células na mesma área de manufatura (fabricação, usinagem ou montagem) ligadas por um sistema de manipulação de material e controladas por computador.

4) **Sistema (SMF)**

Constituído de vários grupos que podem desenvolver diferentes tipos de tarefas.

## IV.3 – SCHEDULING DE SMF

Nesta seção trataremos dos dois principais problemas que surgem ao trabalharmos com o scheduling em SMF: rota de máquinas apropriada para cada peça e a melhor seqüência de montagem das peças baseado no trabalho de NAKAMURA e SHINGU (1985).

SMF: Máquinas multi–propósito, sistema de transporte automato, controle do computador é a definição ao SMF usada por NAKAMURA e SHINGU.

Oferece aumento de produtividade na fábrica que produz uma série de produtos similares em lotes muito pequenos para justificar o não uso de maquinaria de produção em massa tal como transfer lines.

**Problemas**

$1^{o}$. projetar o melhor sistema adaptado com o ambiente de produção (não será considerado aqui).

$2^{o}$. controle do fluxo de peças de modo a satisfazer mais efetivamente os objetivos de produção.

Problemas de scheduling de job shops não automatos são tratados na suposição que a designação de operações para máquinas são fixadas como pré–determinadas devido a não versatilidade das máquinas e do sistema de transporte que requer o uso de controle manual (humano).

No caso do SMF, máquinas são bastante versáteis e capazes de realizar muitos tipos diferentes de operações. Esta capacidade em conjunto com transporte automato pode permitir variedade de padrões de fluxos, podendo existir rotas alternativas.

No algoritmo de Nakamura e Shingu $1^{o}$. estágio mostra um procedimento para selecionar uma rota de máquinas apropriadas. No $2^{o}$. estágio é proposto um procedimento que objetiva definir uma seqüência de montagem minimizando a duração periódica do tempo de montagem.

O sistema aqui tratado é um tipo de SMF que poderia ser chamado "loop–flow". As máquinas são dispostas ao redor de um sistema de transporte fixado em loop.

O transportador leva peças de máquinas para máquina em direções fixas e também tem a função como entrada do buffer.

Fig. IV.1 – SMF tipo loop

Máquinas 1 e K são estações de montagem e desmontagem.

Qualquer outra máquina K = 2, ..., K–1 do sistema é suposta ser capaz de realizar diferentes operações. Cada uma das máquinas tem um armazém de entrada de capacidade ilimitada e estes armazéns irão ser supostos operando segundo a regra FIFO. (O primeiro a entrar é o primeiro a sair).

Cada máquina pode realizar apenas uma operação de cada vez. Suporemos também que os tempos de setup podem ser negligenciados.

Conjunto de mínimo de peças (CMP).

Iremos assumir que as peças i, i = 1, 2, ..., M são produzidas nas razões $r_i$ ($= n_i/N$) onde N é um número pequeno razoável e $n_i$ é o número mínimo de peças i que preenchem as condições que seguem:

$$n_i = r_i \sum_{i=1}^{M} n_i = r_i\, N, \quad i = 1, 2, \ldots, M$$

e

$$\text{m.d.c.}\, \{n_1, n_2, \ldots, n_M\} = 1$$

É então possível definir um conjunto mínimo de peças (CMP) como um conjunto de inteiros

$$E = \{n_1, n_2, \ldots, n_M\}$$

Roteamento das máquinas.

Devido à versatilidade das máquinas o sistema pode realizar diferentes tipos de operações. Esta capacidade permite uma flexibilidade considerável nos padrões de roteamento das máquinas.

$\ell = 1, 2, \ldots, L_i \rightarrow$ índice de roteamento das máquinas da peça i. Devemos selecionar uma rota apropriada entre as alternativas para cada peça.

Seqüência de montagem.

Seja $E^q$, $q = 1, 2, \ldots, Q$ e CMP do q–ciclo e sejam $s_{ik}^q$ e $f_{ik}^q$ os tempos de início e fim respectiva na máquina k para a peça i que pertence ao conjunto $E^q$.

É suposto que as peças são montadas de acordo com seus índices em $E^q$.

Isto implica que:

$$S^q_{1k} < S^q_{1k} < \dots S^q_{MK} \qquad k = 1, 2, \dots, K \quad q = 1, 2, \dots, Q$$

Uma seqüência de montagem é chamada "periódica" com duração CT se

$$S^{q+1}_{ik} = S^q_{ik} + CT \qquad i = 1, \dots, M \quad k = 1, 2, \dots, K$$

CT associado com $E^q$ definido como q = 1, 2, ..., Q–1.

$$CT = \max_x \left(f^q_{ult,k} - s^q_{prim,k}\right)$$

$s^q_{prim,k}$: tempo de início da primeira peça de $E^q$ na máquina k

$f^q_{ult,k}$: tempo de término da última peça de $E^q$ na máquina k.

Chamaremos CT de tempo de ciclo e adotaremos a minimização deste tipo como uma medida de desempenho para nosso problema de scheduling.

Definição do problema:

Dada uma descrição do SMF (número de máquinas e tempo de viagem entre elas) e a produção desejada (número de tipos e peças, número de padrões de roteamento das máquinas, tempos de operação nas máquinas e as desejadas razões de produção), determine a rota das máquinas e a seqüência de montagem pelas quais as peças serão produzidas de modo que o tempo de ciclo do CMP seja tão pequeno quanto possível.

Enfoque heurístico:

$p^{\ell}_{ik}$: tempo de operação da peça i na máquina k na rota $\ell$.

Para $n_i$ peças em um CMP

$t^{\ell}_{ik} = n_i \, p^{\ell}_{ik}, \quad i = 1, ..., M \quad k = 1, ..., K \quad \ell = 1, ..., \ell_i$

$p_{ik}$: tempo de operação da peça i na máquina k quando uma rota particular é selecionada.

Para $n_i$ peças em um CMP

$t_{ik}$: $n_i \, p_{ik}$, $i = 1, 2, ..., M$, $k = 1, 2, ..., K$

$v^{\ell}_{km}$: tempo de viagem da máquina k para máquina m na rota $\ell$.

Seleção da rota:

**Objetivo**

Selecionar uma rota apropriada para cada peça, baseado na minimização do tempo de ciclo.

Não consideraremos inicialmente a seqüência das peças. Então o tempo de ciclo para um CMP é equivalente à máxima carga de trabalho expresso com uma unidade de tempo, que pode ser efetuado pela máquina gargalo para um CMP.

Suponhamos que a rota $\in$ para a peça i é selecionada. A carga de trabalho $T^a$ na máquina k é:

$$T_k = \sum_i t_{ik} \qquad k = 1, \ldots, K$$

e carga de trabalho máximo $T_{max}$ é

$$T_{max} = \max_k \{T_k\}$$

Então cada máquina, exceto a máquina com $T_{max}$ tem a perda de equilíbrio $(T_{max} - T_k)$.

Suponhamos que a rota $\in$ é trocada com a rota $\ell$. Então, carga de trabalho na máquina k é mudado por $(t_{ik}^{\ell} - t_{ik})$.

Definimos o índice:

$$d_{ik}^{\ell} = t_{ik}^{\ell} - t_{ik} - (T_{max} - T_k)$$

$$i = 1, \ldots, M \qquad k = 1, \ldots, K \qquad \ell = 1, 2, \ldots, \ell_i$$

Podemos escrever na forma matricial:

$$D = \begin{bmatrix} d_{1\,1}^{1} & d_{1\,2}^{1} & \ldots & d_{1\,k}^{1} \\ d_{1\,1}^{2} & d_{1\,2}^{2} & \ldots & d_{1\,k}^{2} \\ \vdots & & & \\ d_{MK}^{LM} & d_{MK}^{LM} & \ldots & d_{MK}^{LM} \end{bmatrix}$$

Este índice representa a diferença de carga de trabalho na máquina k que ocorre pela troca $\epsilon$ com $\ell$ para peça i.

Então podemos usar a matriz de índice para determinar quando esta troca deve ser feita ou não.

Seja:

$$d = \min_{i} \{\max_{k} d_{ik}^{\ell}\}$$

Se $d < 0$, então seja $\ell =$ a com $d < 0$, troque $\ell$ com n.

Desse modo esta troca irá levar a uma redução em $T_{max}$.

Neste processo de troca, nós necessitamos determinar o valor inicial (rota máquinas inicial) para cada peça.

Definimos o seguinte tempo de fluxo:

$$F_i^{\ell}: \sum_{k} (t_{ik}^{\ell} + v_{km}^{\ell}) \qquad i = 1, ..., M \qquad \ell = 1, 2, ..., \ell_i$$

Podemos usar a rota com $\min_{\ell} F_i^{\ell}$ para peça i como a rota inicial.

Decisão de seqüência de montagem:

Determina a rota no estágio anterior e queremos agora obter a seqüência de montagem.

Para o FMS pode ser difícil obter a seqüência ótima, devido à complexidade. Definiremos então uma heurística que permite obter uma melhor seqüência de montagem para um CMP.

Este procedimento é baseado na seguinte heurística: 1º. gerar uma seqüência na máquina base, depois fazer o schedule das peças nas outras máquinas e selecionar o schedule que tem os conflitos mínimos. Escolheremos a máquina gargalo como a máquina base.

Calcularemos o conflito entre peças, baseado no tempo de início $s_{ik}$ e tempo de término $f_{ik}$ da peça i na máquina base h é possível definir o tempo inicial $s_{ik}$ e tempo de término $f_{ik}$ da peça i em qualquer máquina k.

$$s_{ik} = \begin{cases} s_{ih} - \sum\limits_{\alpha} (p_{i\alpha} + v_{\alpha\beta}) & (O_{ik} < O_{ik}) \\ s_{ih} + \sum\limits_{\alpha} (p_{i\alpha} + v_{2\beta}) & (O_{ik} > O_{ih}) \end{cases}$$

$$f_{ik} = s_{ik} + p_{ik}$$

$O_{ik}$ – é a operação da peça i na máquina k.

Um índice de conflito $c_k$ $(i < j)$ entre peças i e j pode ser definido na máquina k:

$$c_k\ (i < j) = \max[0, \min(f_{ik}, f_{jk}) - \max(s_{ik}, s_{jk})] \quad k = 1, 2, ..., K$$

Se uma das peças i ou j, ou as duas não usam a máquina k então $c_k(i < j) = 0$.

Então um conflito global entre peças i e j para todas as máquinas pode ser definido por

$$C(i < j) = \sum_k c_k(i < j)$$

Para uma seqüência parcial $S_G$ de G peças na máquina base, o conflito parcial $S(S_g)$ pode ser definido por

$$C(S_G) = \sum_{j=2}^{G} \sum_{i=1}^{j-1} C(i < g)$$

Para obter um schedule viável, devemos acabar com o conflito.

**Algoritmo de Nakamura e Shingu**

**Seleção da Rota**

**Passo 1:** rota $\ell = \epsilon$ para peça i tal que $F_i^{\ell}$ é o mínimo.

**Passo 2:** calcular $T_k$ e $T_{max}$. Vá para 6.

**Passo 3:** calcular matriz D.

**Passo 4:** busca d para todas as peças e máquinas.

**Passo 5:** se existe rota n tal que $d < 0$ então $\ell = n$! troca $\epsilon$ com n e volta 2.

Caso contrário vá para 12.

**Decisão da Seqüência de Montagem**

**Passo 6:** define máquina gargalo ($T_{max}$) como a máquina base e escolhe a primeira peça da seqüência naquela máquina dentre os M tipos de peças.

**Passo 7:** gera a subseqüência $S_G$ na máquina base h pela adição de nova peça e calcula $C(S_G)$ para cada nova subseqüência gerada. (Se existe peça que não é processada na máquina base, podemos introduzir uma peça cujo tempo de operação é zero).

**Passo 8:** busca a subseqüência que tem $C(S_G)$ mínimo.

**Passo 9:** Se $G = M$, vá para 10. Caso contrário vá para 7.

**Passo 10:** dissolve o conflito no schedule de conflito mínimos selecionados no passo 9.

**Passo 11:** obtém o tempo de ciclo CT e volte para 3.

**Passo 12:** busca o schedule em que CT é mínimo e obtém a rota e seqüência de montagem com CT mínimo. Pare.

Exemplo:

| CASO | CONFLITOS | PRECEDÊNCIA | PROCEDIMENTO | RESULTADO |
|---|---|---|---|---|
| 1 | máq.k B C<br>máq. h A B C | $0_{Bk} < 0_{Bh}$<br>$0_{Ck} < 0_{ch}$ | faixa c e coloque<br>B antes de C | K B C<br>h A B C |
| 2 | k A B<br>h A B C | $0_{Ak} > 0_{h}$<br>$0_{Bk} > 0_{BH}$ | fixa A e coloca<br>B depois de A | k A B<br>n A B C |
| 3 | k C A<br>h A B C | $0_{Ak} > 0_{Ah}$<br>$0_{Ck} < 0_{Ch}$ | fixa A e coloca C<br>antes de A ou<br>fixa C e coloca A<br>depois de C | k C A<br>l A B C<br>k C A<br>h A B C |

Número de máquinas: k = 6

Número de tipos de peças: M = 4

Conjunto mínimo de peças: E = {1, 1, 1, 4}

**Rotas**

| | | OPERAÇÃO | | | | | |
|---|---|---|---|---|---|---|---|
| **PEÇA** | **ROTA** | **1** | **2** | **3** | **4** | **5** | **6** |
| 1 | 1 | $M_1$ | $M_2$ | $M_3$ | $M_4$ | $M_5$ | $M_6$ |
| | 2 | $M_1$ | $M_4$ | $M_3$ | $M_2$ | $M_5$ | $M_6$ |
| | 3 | $M_1$ | $M_3$ | $M_2$ | $M_5$ | $M_4$ | $M_6$ |
| 2 | 1 | $M_1$ | $M_5$ | $M_2$ | $M_4$ | $M_3$ | $M_6$ |
| | 2 | $M_1$ | $M_3$ | $M_4$ | $M_2$ | $M_5$ | $M_6$ |
| | 3 | $M_1$ | $M_4$ | $M_2$ | $M_5$ | $M_3$ | $M_6$ |
| 3 | 1 | $M_1$ | $M_3$ | $M_2$ | $M_5$ | $M_4$ | $M_6$ |
| | 2 | $M_1$ | $M_2$ | $M_4$ | $M_5$ | $M_3$ | $M_6$ |
| | 3 | $M_1$ | $M_4$ | $M_3$ | $M_5$ | $M_2$ | $M_6$ |
| 4 | 2 | $M_1$ | $M_4$ | $M_2$ | $M_3$ | $M_5$ | $M_6$ |
| | 3 | $M_1$ | $M_4$ | $M_5$ | $M_3$ | $M_2$ | $M_6$ |

**Tempos de Operação**

| PEÇA | OPERAÇÃO 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 2 | 4 | 7 | 5 | 4 | 2 |
| 2 | 2 | 7 | 6 | 4 | 5 | 2 |
| 3 | 2 | 5 | 3 | 3 | 4 | 2 |
| 4 | 2 | 3 | 9 | 5 | 6 | 2 |

**Tempos de viagem**

| PARA/DE | $M_1$ | $M_2$ | $M_3$ | $M_4$ | $M_5$ | $M_6$ |
|---|---|---|---|---|---|---|
| $M_1$ | 5 | 4 | 3 | 2 | 1 | 1 |
| $M_2$ | 1 | 5 | 4 | 2 | 2 | 1 |
| $M_3$ | 2 | 1 | 5 | 4 | 3 | 2 |
| $M_4$ | 3 | 2 | 1 | 5 | 4 | 3 |
| $M_5$ | 4 | 3 | 2 | 1 | 5 | 4 |
| $M_6$ | 2 | 1 | 5 | 4 | 3 | 2 |

a) rota

| PEÇA | | ROTA |
|---|---|---|
| 1 | 1: | $M_1 - M_2 - M_3 - M_4 - M_5 - M_6$ |
| 2 | 2: | $M_1 - M_3 - M_4 - M_2 - M_5 - M_6$ |
| 3 | 2: | $M_1 - M_2 - M_4 - M_5 - M_3 - M_6$ |
| 4 | 1: | $M_1 - M_3 - M_2 - M_4 - M_5 - M_6$ |

b) seqüência de montagem

| MÁQUINA | SEQÜÊNCIA DE MONTAGEM |
| --- | --- |
| 1 | 2( 0) – 3( 9) – 1(12) – 4(14) |
| 2 | 3(12) – 1(17) – 2(21) – 4(25) |
| 3 | 2( 4) – 4(18) – 1(22) – 3(29) |
| 4 | 2(12) – 3(19) – 1(30) – 4(36) |
| 5 | 3(23) – 2(28) – 1(36) – 4(42) |
| 6 | 2(34) – 3(36) – 1(41) – 4(49) |

Número entre parênteses são os tempos de início.

c) **Tempo de ciclo:** CT = 29

## IV.4 – SEQENCIAMENTO DOS PLANOS DE PROCESSAMENTO

### IV.4.1 – Introdução

Na seção anterior apresentamos uma heurística desenvolvida por NAKAMURA & SHINGU para o problema de seqüenciamento de montagem das peças. Nesta seção faremos uso do Problema do Caixeiro Viajante que nos permitirá obter uma solução bastante satisfatória do problema, senão a ótima.

Assumiremos aqui que o problema representado pela escolha dos planos de processamento que passaremos sinteticamente a delinear está resolvido. Sugerimos também, para maiores referências o trabalho de COCA–BALTA (1991). Um plano de processamento para uma peça pode ser definido como um conjunto de operações elementares que são necessários para a produção desta peça. Por operações elementares entenda–se a operação ou tarefa de retirada de um volume de matéria prima para que se possa obter uma dada peça P. Esses volumes são retirados através de corte, furo ou alguma ação das máquinas individualmente ou por grupos através de uma operação elementar. De acordo com a disponibilidade de máquinas e ferramentas devem ser feitas várias operações elementares para cada peça.

Este plano de processamento determina então a seqüência de visitação às máquinas que deve ser feita para a produção de uma dada peça.

Definidos então os planos de processamento de cada peça desejada passemos então a definir a seqüência em que deverão ser produzidas tais peças.

**IV.4.2 – Problema de Seqüência dos Planos de Processamento**

Fundamentaremos esta seção à partir de estudo feito por COCA–BALTA (1991).

Deseja–se produzir m tipos de peças diferentes sendo que a peça i é produzida pelo plano de processamento $P_i$, $i = 1, ..., m$.

Ao passarmos da produção de uma peça para outra determinadas máquinas ou ferramentas podem ser trocadas para que se possa usinar a peça seguinte. Caracteriza–se então o setup que produzirá um custo de troca, ao se passar da produção de uma peça para a seguinte.

Para se definir estes custos de setup consideremos que:

Os planos de processamento $P_i$ e $P_j$ irão produzir as peças i e j respectivamente, havendo um custo $c_{ij}$ decorrente do setup necessário para a execução do plano $P_j$ logo após a execução de $P_i$. Devemos levar em conta as máquinas e as ferramentas que foram usadas no plano $P_j$ para se obter $c_{ij}$.

Define-se uma matriz $B = (b_{ki})$ de n linhas e m colunas sendo que as linhas representam as ferramentas ou máquinas disponíveis e as colunas representam os planos de processamento que correspondem aos tipos de peças a serem produzidas.

Cada elemento $b_{ki}$ da matriz define-se como:

$$b_{ki} = \begin{cases} 1 \text{ se a ferramenta ou máquina k é usada no plano } P_i \\ 0 \text{ caso contrário} \end{cases}$$

A exigência para que uma dada ferramenta seja usada em um novo plano de processamento pode ser expressa por:

$$\delta(b_{ki}, b_{kj}) = \begin{cases} 1 \text{ se } b_{ki} \neq b_{kj} \\ 0 \text{ caso contrário} \end{cases}$$

Podemos ter dois casos em que $b_{ki} \neq b_{kj}$:

1) se $b_{ki} = 1$ e $b_{kj} = 0$ não há necessidade da ferramenta ou máquina k no plano $P_j$ devendo ser retirada após seu uso para podermos iniciar o plano $P_j$ (usinagem da peça j).

2) se $b_{ki} = 0$ e $b_{kj} = 1$ devemos montar a ferramenta ou a máquina k pois esta não estava sendo usada no plano $P_i$.

Define-se portanto os custos de troca através de:

$$c_{ij} = \sum_{k=1}^{n} \lambda_k * \delta(b_{ki}, b_{kj})$$

sendo $\lambda_c$ correspondente aos custos de troca de cada ferramenta k.

Definimos anteriormente que o PCV deve percorrer um dado conjunto de cidades, cada cidade uma única vez minimizando–se a distância percorrida.

Pois nosso problema de seqüenciamento dos planos de processamento traz em si todas essas condições ao procurar a otimalidade da solução, minimizando os custos de setup. Vejamos:

i) busca–se o menor custo de setup (custo de troca) o que equivale à minimização da distância percorrida após a visitação das cidades;

ii) todos os planos devem ser excutados uma única vez equivalente a se percorrer todas as cidades uma única vez;

iii) os custos de troca $c_{ij}$ decorrentes das mudanças requeridas do plano $P_i$ para o plano $P_j$ corresponde à distância entre as cidades i e j no PCV.

Percebe–se então claramente que pode–se formulá–lo como um Problema Caixeiro Viajante.

O problema de seqüenciamento pode ser formulado através do PCV como:

$$\text{minimizar} \sum_{i=0}^{m} \sum_{\substack{j=1 \\ j \neq i}}^{m+1} c_{ij} + x_{ij}$$

sujeito a:

$$\sum_{\substack{i=0 \\ i \neq j}}^{m} x_{ij} = 1, \qquad j = 1, ..., m+1 \tag{IV.1}$$

$$\sum_{\substack{j=1 \\ i \neq j}}^{m+1} x_{ij} = 1, \qquad i = 0, ..., m \tag{IV.2}$$

$$\sum_{i \in S} \sum_{j \in \bar{S}} x_{ij} \geq 1, \qquad S \subset P = \{P_1, ..., P_m\}, S \neq 0 \tag{IV.3}$$

$$\bar{S} = P - S$$

$$x_{ij} \in \{0, 1\}$$

Considerando que a variável de decisão $x_{ij}$ significa:

$$x_{ij} = \begin{cases} 1 \text{ se os planos } P_i \text{ e } P_j \text{ são consecutivos} \\ 0 \text{ caso contrário} \end{cases}$$

As restrições (IV.1) e (IV.2) tem funções equivalentes às prestadas no PCV: garantem que uma vez o plano iniciado este é terminado uma única vez.

Considerando-se que para se processarem determinadas peças algumas operações devem necessariamente ser feitas precedendo-se outras faz-se necessário acrescentar no modelo apresentado restrições de precedência entre as tarefas, o que seria equivalente a se exigir que algumas cidades sejam visitadas antes que outras na rota do caixeiro.

Uma solução para o acréscimo destas restrições é seguir o sugerido por [COCA-BALTA, 1991] e usar a formulação do Caixeiro Viajante como um modelo de fluxo em rede de dois produtos [FINKE, CLAUS & GUNN, 1984] visto no Capítulo I. Claramente, o acréscimo destas restrições de precedência irá aumentar a complexidade do problema.

**Notação**: Usaremos $<$ para definir a relação de precedência.

Logo, se $P_j$ deve ser feito após $P_i$ tem–se $P_i < P_j$.

São consideradas m tarefas e também uma tarefa extra (fictícia) que representa o estágio inicial onde não há ferramentas ou máquinas montadas o que caracteriza o setup vazio ou inicial. O plano $P_0$ equivale a essa tarefa, em que não há necessidade de ferramentas, portanto há um total de m+1 tarefas.

Consideremos que os fluxos $x_{ij}$ e $y_{ij}$ representam o número de nós que devem ser analisados e os que já foram, respectivamente.

No caso de termos $P_u < P_v$ o fluxo $x_{uj}$ deverá ser pelo menos uma unidade maior que o fluxo $x_{vj}$ pois só poderemos iniciar $P_v$ se já tiver processado $P_u$.

O modelo será então formulado como:

$$\text{minimizar } \frac{1}{m} \sum_i \sum_j (x_{ij} + y_{ij}) * c_{ij}$$

sujeito a:

$$\sum_{j=0}^{m} x_{ij} - \sum_{j=0}^{m} x_{ji} = p_i \qquad i = 0, \dots, m$$

$$\sum_{j=0}^{m} y_{ij} - \sum_{j=0}^{m} y_{ji} = q_i \qquad i = 0, 1, \dots, m$$

$$x_{ij} \geq 0,\ y_{ij} \geq 0 \quad i = 1, \ldots, m \text{ e } j = 0, 1, \ldots, n$$

$$\sum_{j=0}^{m} (x_{ij} + y_{ij}) = m \qquad i = 0, 1, \ldots, m$$

$$x_{ij} + y_{ij} \in \{0, m\} \qquad i = 0, 1, \ldots, m \text{ e } j = 0, 1, \ldots, n$$

$$\sum_{j} x_{uj} - \sum_{j} y_{vj} \geq 1 \qquad i = 1, \ldots, m$$

## IV.5 – CUIDADOS NA FORMULAÇÃO DO PCV AO SE RESOLVER PROBLEMAS PRÁTICOS

CARTER, MAGAZINE e MOON (1988) no paper "A Warning About Cyclic Cot Scheduling With Sequence Dependent Setup Times" fazem algumas observações importantes em relação ao problema de scheduling de lotes cíclicos. Este problema se caracteriza, segundo eles, por: tamanho dos lotes de N produtos diferentes são disponíveis para processamento em uma técnica máquina; troca de um tipo de produto para outro acarreta custos de setupr que dependem de ambos os tipos de produtos. Temos que o tempo de processamento de cada produto (lead time) não depende da ordem em que eles são produzidos, portanto **minimizar o tempo de processamento total** é equivalente a **achar a seqüência de produtos que minimiza o custo total de setup.**

Outros autores que trabalharam com o problema de minimização de tempos de setup em modelos de produção em lote foram OZDEN et al. (1985); FAO e WAGNER (1983); WHITE e WILSON (1977). Em todos estes trabalhos os autores fazem uma suposição implícita de que o tamanho dos lotes não pode ser dividido. É com relação à esta suposição que CARTER et al. apresentam um contra–exemplo.

Nos papers acima citados os autores estabelecem que o problema de tempo de setup dependente do sequenciamento é equivalente ao problema do caixeiro viajante (PCV) onde os produtos representam as cidades e o tempo de setup entre dois produtos é equivalente à distância entre as cidades correspondentes.

Consideremos o seguinte exemplo, onde quatro produtos são produzidos em ambiente ou lote.

Os tempos de setupr, $s_{ij}$, para início do lote do produto j imediatamente depois do produto i são dados abaixo:

| | j | | | |
|---|---|---|---|---|
| i | 1 | 2 | 3 | 4 |
| 1 | 0 | 1 | 1 | 1 |
| 2 | 1 | 0 | 10 | 10 |
| 3 | 1 | 10 | 0 | 10 |
| 4 | 1 | 10 | 10 | 0 |

O problema é determinar o schedule de produçÕ cíclica que minimiza o tempo de setup total. Claramente tem-se que qualquer rota representando uma solução viável para o PCV terá um tempo total de setup igual a 22. Observa-se, entretanto, que a seqüência [1, 2, 1, 3, 1, 4] tem um tempo de setup total de 6 por ciclo. Este plano pode ser implementado dividindo-se o tamanho do lote do produto 1 em 3 lotes por ciclo. O PCV assume que cada cidade deve ser visitada exatamente uma vez. Ao se relaxar esta restrição e permitindo que cada cidade possa ser visitada ao menos uma vez, nós poderíamos achar soluções de custo menor. A desvantagem que lotes pequenos usualmente trazem é o aumento no número de setups. Se reduzirmos os lotes e diminuir os tempos de setup, podemos obter uma vantagem adicional de diminuir os custos de inventário.

O problema relaxado não irá produzir custos menores à menos que a desigualdade triangular seja violada. Esta condição assume que para quaisquer 3 produtos A, B e C, o tempo de setup do produto A para o produto C nunca é maior do que o tempo total de A para B para C. FOO e WAGNER (1983) usam um exemplo tirado de produção real que contém várias violações.

Pode–se transformar o problema de scheduling destes cíclicos exibidos aqui em um PCV equivalente. Se substituimos os tempos de setup, $s_{ij}$, em cada aresta pelo custo do caminho mínimo para se ir de i para j, teremos que o novo problema irá satisfazer a desigualdade triangular. No exemplo o tempo de setup da tarefa 1 depois da tarefa 3 irá se tornar 2 (caminho 3, 1, 2). O custo de uma rota ótima [1, 2, 3, 4, 1], por exemplo, irá ser 6 mas as arestas [2, 3, 4] implicam uma seqüência de setup [2, 1, 3, 1, 4].

Após ver o estudo acima, fica então claro que problemas deste tipo implicam num maior cuidado do que simplesmente formulá–lo com um PCV padrão.

# CAPÍTULO V

# CONCLUSÃO

O problema do Caixeiro Viajante (PCV) por sua reconhecida importância teórica, avaliada pelo elevado número de publicações na literatura afim, encontra também em nossa dissertação, mais uma aplicação para problemas de ordem prática.

Definimos a estrutura deste trabalho em duas partes: a apresentação do PCV com diversas modelagens, algumas das quais poderiam refletir melhor a estrutura dos problemas de manufatura flexível, já que no caso do PCV padrão, a suposição mais assumida na literatura é a obediência à desigualdade triangular. Esta suposição poderia "mascarar" algumas soluções, pois apresentamos um exemplo de ordem prática que deixa evidente tal situação.

As principais estratégias de resolução são apresentadas e separadas dentre métodos exatos e aproximados. Conforme citado anteriormente os métodos mais utilizados são os aproximados mas isto não invalida a importância dos métodos exatos, pois muitas das heurísticas são formuladas intuitivamente através da estrutura dos problemas que foram exaustivamente esmiuçados pelos estudos feitos para os métodos exatos.

Na segunda parte do trabalho apresentamos os conceitos de manufatura flexível e alguns problemas existentes. Uma resolução do problema de seqüenciamento dos Planos de Processamento das peças é apresentado e faz uso do Problema do Caixeiro Viajante. Este problema, apesar de sua reconhecida complexidade, poderia em certos casos, a partir de uma análise dos custos/benefícios fornecer uma solução tão satisfatória que compense as dificuldades inerentes ao problema.

Também apresentamos uma heurística formulada por Nakahura e Shingu, que se garante uma solução ótima pode produzir resultados aproximados bastante satisfatórios.

Finalmente gostaríamos de enfatizar que este estudo é apenas o começo de uma pesquisa. Poderia ser desenvolvida, envolvendo uma implementação de alguns métodos de resolução do PCV e testados alguns exemplos práticos de problemas de SMF.

## REFERÊNCIAS BIBLIOGRÁFICAS


AGOSTINHO, O. L. (1985) – "Estudo da Flexibilidade dos Sitemas Produtivos", Tese Dr. em Eng., Escola de Engenharia de São Carlos, USP, SP.

BALAS, E. (1968) – "A Note on the Branch and Bound Principles", **Operations Research 16**, 442–445.

BALAS, E. (1971) – Intersection Cuts – A New Type of Cutting–Plane for Integer Programming", **Operations Research 19**, 19–30.

BALAS, E. and CHRISTOFIDES, N. (1981) – "A Restructed Lagrangean Approach to the Traveling Salesman Problem", **Math. Prog. 21**, 19–46.

BALAS, E. and CHRISTOFIDES, N. (1976) – "A New Penalty Method for the Traveling Salesman Problem", presented at the 9th **Math. Prog. Symposium**, Budapest.

BALAS, E. and GUIGNARD, M. (1979) – "Branch and Bound/Implicit Enumeration", **Ann. Discrete Math., 5**, 185–191.

BRASKARAN, K. (1990) – "Process Plan Selection", **Int. J. Prod. Res., 28(8)**, 1527–1539.

BELLMAN, R. E. (1962) – "Dynamic Programming Treatment of the Traveling Salesman Problem", **J. Assoc. Comp. Mach., 9**, 61–63.

BELLMORE, M. and MALONE, J. C. (1971) – "Pathology of Traveling Salesman

Subtour Elimination Algorithms", **Oper. Res., 19**, 278–307.

BODIN, L. D., GOLDEN, B. L., ASSAD, A. and BALL, M. (1983) – "Routing and Scheduling of Vehicles and Crews, the State of the Art", **Computers and Operations Research, 10**, 69–211.

CHRISTOFIDES, N. (1975) – "Graph Theory – An Algorithmic Approach", Academic Press, Londres.

CHRISTOFIDES, N. (1979) – "The Traveling Salesman Problem, CHRISTOFIDES, N., MINGOZZI, A., TOTH, P. and SANDI, C. editors, **Combinatorial Optimization**, Wiley, Chichester 131–149.

CLARKE, G. and WRIGHT, J. W. (1964) – "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points", **Oper. Res., 12**, 568–581.

CLAUS, A. (1984) – "A New Formulation for the Traveling Salesman Problem", SIAM Journal of **Algebraic Discrete Methods, 5**, 21–25.

COCA–BALTA, A. (1991) – "Modelos de Otimização para Manufatura Automatizada", Tese de D.Sc., COPPE/UFRJ.

EDMONDS, J. (1967) – Optimum Branchings", **J. Res. Nat. Bur. Standards 71B**, 233–240.

FINKE, G., CLAUS, A. and GUNN, E. (1984) – "A Two–Commodity Network Flow Approach to the Traveling Salesman Problem", **Congressus Numerantium 41**, 167–178.

FLOYD, R. (1964) – "Algorithm 245: Treesort 3", Commum. ACM 7(12), 701.

FOX, K. – "Production Scheduling on Parallel Lines with Dependences", Ph.D. Thesis, John Hopkins University (Baltimore, MD).

FOX, K., GAVISH, B. and GRAVES, S. (1990) – "An n–Constraint Formulation of the (Time–Dependent) Traveling Salesman Problem", **Oper. Res., 28,** 1018–1021.

FULKERSON, D. R. (1974) – "Packing Rooted Cuts in a Weighted Directed Graph", **Math. Programming, 6,** 1–13.

GAREY, M. R. and JOHNSON, D. S. (1979) – "Computers and Intractability: A Guide to the Theory of NP Completeness", Freeman, San Francisco.

GARFINKEL, R. S. (1979) – "Branch and Bound Methods for Integer Programming", on **Combinatorial Optimization,** CHRISTOFIDES, N., MINGOZZI, A., TOTH, P. and SANDI, C. editors, Wiley, New York.

GARFINKEL, R. S. and NEMHAUSER, G. L. (1972) – **Integer Programming,** Wiley, New York.

GEOFFRION, A. M. (1974) – "Lagrangean Relaxation for Integer Programming", **Math. Programming Stud. 2,** 82–114.

GEOFFRION, A. M. and MARSTEN, R. E. (1972) – Integer Programming Algorithms: A Framework and State of the Art Survey", **Management Science**

**18**, 465–491.

GILLETT, B. E. and MILLER, L. R. (1974) – "A Heuristic Algorithm for the Vehicle–Dispatch Problem", Oper. Res. 22, 341–349.

GOLDEN, B. (1977) – "A Statistical Approach to the TSP", Networks 7, 209–225.

GOMORY, R. E. (1960) – "Solving Linear Programming Problems in Integers", **Proc. Sympos. Appl. Math. 10**, 211–215.

GOMORY, R. E. (1963) – "An Algorithm for Integer Solutions to Linear Programs", GRAVES, R. L. and WOLFE, P. editors, **Recent Advances in Mathematical Programming**, McGraw–Hill, New York, 269–302.

GOMORY, R. E. (1966) – "The Traveling Salesman Problem", **Proc. IBM Scientific Computing Symposium Combinatorial Problems**, IBM Data Processing Division, White Plains, New York.

HARTLEY, J. (1984) – "FMS at Work", IFS Ltd. UK., North–Holland.

HELD, M. and KARP, R. M. (1962) – "A Dynamic Programming Approach to Sequencing Problems", **SIAM J. Appl. Math. 10**, 196–210.

HELD, M. and KARP, R. M. (1970) – "The Traveling Salesman Problem and Minimum Spanning Trees", **Oper. Res. 18**, 1138–1162.

HELD, M. and KARP, R. M. (1971) – "The Traveling Salesman Problem and Minimum Spanning Trees II", **Math. Programming 1**, 6–25.

HWAN, S. S. and SHOGAN, A. W. (1989) – "Modelling and Solving an FMS Part Selection Problem", **Int. J. Prod. Res. 27(8)**, 1349–1366.

KUSIAK, A. (1985) – "Flexible Manufacturing System: A Structural Approach", Int. **J. of Prod. Res. 23(6)**, 1057–1073.

KUSIAK, A. and FINKE, G. (1988) – "Selection of Process Plans in Automated Manufacturing Systems", **IEEE J. of Robotics and Automation 4(4)**, 397–402.

LAND, A. and POWELL, S. (1977) – "Computer Codes for Problems of Integer Programming", **presented at the Conference on Discrete Optimization**, Vancouver, B.C.

LAWLER, E. L., LENSTRA, J. K., RINNOOYKAN, A. H. and SHMOYS, D. B. (eds.) (1985) – "The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization", Wiley, Chichester.

LIN, S. (1965) – "Computer Solutions of the Traveling Salesman Problem", **Bell System Tech. J. 44**, 2245–2269.

LINS, S. and KERNIGHAN, B. W. (1973) – "An Effective Heuristic Algorithm for the Traveling Salesman Problem", **Oper. Res. 21**, 498–516.

LITTLE, J. D. C., MURTY, K. G., SWEENEY, D. W. and KAREL, C. (1963) – "An Algorithm for the Traveling Salesman Problem", **Oper. Res. 11**, 972–989.

MACULAN, N., CAMPELO, R. E. e LOPEZ, L. B. R. (1984) – "Relaxação

Lagrangeana em Programação Inteira", COPPE/UFRJ, Relatório Técnico ES 40–84, Rio de Janeiro.

MILLER, C., TUCKER, A. and ZEMLIM, R. (1960) – "Integer Programming Formulations and Traveling Salesman Problem", **Journal of the Association for Computing Machinery 7**, 326–329.

MINOUX, M. (1986) – "Mathematical Programming Theory and Algorithms", Wiley.

NAKAMURA, H. and SHINGU, T. (1985) – "Scheduling of Flexible Manufacturing Systems", Proceedings of the 8th Int. Conf. on Prod. Res. and 5th Working Conf. of the Fraunhofer – Institute for Industrial Engineering (FHG–IAO), University of Stuttgart, BULLINGER, H. J. and WARNECE, H. J. editors.

NEMHAUSER, G. L. and WOLSEY, L. A. (1988) – "Integer and Combinatorial Optimization", Wiley.

ONG, H. (1981) – "Design and Analysis of Heuristics for Some Routing and Packing Problems", Ph.D. Thesis, Universidade de Waterloo.

PADBERG, M. and SUNG, T. Y. (1991) – "An Analytical Comparison of Different formulations of the Traveling Salesman Problem", **Math. Programming 52**, 315–357.

PAPADIMITRIOU, C. H. and STEIGLITZ, K. (1982) – "Combinatorial Optimization: Algorithms and Complexity, Prentice–Hall, Englewood Cliffs, N.J.

PICARD, J. and QUEYRANNE, M. (1978) – "The Time–Dependent Traveling Salesman Problem and its Application to the Tardiness Problem in One–Machine Scheduling", Oper. Res. 26, 86–110.

ROSENKRANTZ, D. J., STEARNS, R. E. and LEWIS, P. M. (1977) – "An Analysis of Several Heuristics for the Traveling Salesman Problem", **SIAM J. Comp. 6**, 563–581.

ROSENKRANTZ, D. J., STEARNS, R. E. and LEWIS, P. M. (1974) – "Approximate Algorithms for the TSP", Proc. 15th IEEE **Symp. on Switching and Automatic Theory 33.**

SALKIN, H. W. (1975) – "Integer Programming", Addison–Wesley Pub. Co., Reading, Mass.

SMITH, T. H. C. and THOMPSON, G. L. (1975) – "A Comparison of Two Different Lagrangean Relaxation of the TSP", MSR 382, Carnegie–Mellon University.

TAHA, H. (1975) – "Integer Programming Theory Applications and Computations", Academic Press.

WILLIAMS, J. (1974) – Algorithm 232: Heapsort, Commun. ACM 7, 347–348.

WITTROCK, R. J. (1990) – "Scheduling Parallel Machines With Major and Minor Setup Times", **The Int. J. of Flexible Manufacturing Systems 2**, 329–341.