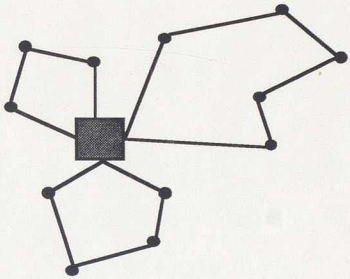
Maria Cândida Vergueiro Monteiro Cidade Mourão



Métodos Aproximativos e Exactos para o Problema do Caixeiro Viajante Múltiplo

um estudo computacional

Departamento de Estatística, Investigação Operacional e Computação
Faculdade de Ciências da Universidade de Lisboa

Junho 1989

Maria Cândida Vergueiro Monteiro Cidade Mourão

Métodos Aproximativos e Exactos para o Problema do Caixeiro Viajante Múltiplo

um estudo computacional

Dissertação destinada à obtenção do grau de Mestre em Estatística e Investigação Operacional na Faculdade de Ciências da Universidade de Lisboa

Departamento de Estatística, Investigação Operacional e Computação
Faculdade de Ciências da Universidade de Lisboa

Junho 1989

Aos meus futuros filhos (*)

(*) O mais velho já teria pelo menos dois anos, se não fosse este trabalho.

Agradecimentos

Pretendo antes de mais e muito sinceramente agradecer à Prof. Teresa Almeida por toda a orientação prestada durante o tempo, quase infindável, de duração deste trabalho.

Um grande obrigada aos meus colegas do I.S.E. que de alguma forma me ajudaram nesta "árdua" tarefa, com um duplo agradecimento à Leonor e ao Alípio.

Também para Ciências vai um beijão aos que me consegiram aturar, sem refilar.

Agradecendo e simultâneamente pedindo desculpa aos meus mais próximos familiares pelo meu mau feitio adicional, fica um beijo muito especial para o Carlos.

Resumo

O problema do Caixeiro Viajante Múltiplo consiste na determinação das rotas óptimas a atribuir a dois ou mais caixeiros, que partindo todos de uma mesma cidade, a ela regressam no final da viagem. Todas as cidades, à excepção da inicial, são visitadas exactamente uma vez por um único caixeiro.

Na presente tese são apresentados métodos aproximativos e exactos para este problema. Após implementados, os diversos métodos foram analisados em termos das soluções que proporcionam. Com base nos resultados obtidos são tiradas algumas conclusões.

Os valores das soluções óptimas de problemas Euclideanos, com um máximo de **30** vértices e **3** caixeiros, permitiram concluir que a heurística implementada revela interesse essencialmente teórico. Foi ainda possível observar, que o método de utilizado na determinação de minorantes proporcionou a obtenção de soluções de valores não muito afastados do valor óptimo do problema.

ÍNDICE

1. Introdução

Nesta tese são estudados e implementados computacionalmente métodos aproximativos e exactos o problema do Caixeiro Viajante Múltiplo.

O problema, colocado de uma forma simples e pragmática, consiste na determinação das rotas óptimas a atribuir a dois ou mais caixeiros, que partindo todos de uma mesma cidade, a ela regressam no final da viagem. Todas as cidades, à excepção da inicial, são visitadas exactamente uma vez por um único caixeiro.

Este problema é uma das generalizações do "bem conhecido" problema do Caixeiro Viajante no qual se considera apenas um caixeiro. São ambos problemas de "difícil" resolução, ou seja, para o quais não existem, e se presume ser impossível desenvolver, algoritmos que no pior caso, determinem a sua solução óptima em tempo de execução polinomial. Tal justifica a importância do desenvolvimento de "bons" métodos aproximativos, ou seja métodos que forneçam soluções de valores não muito afastados do valor óptimo do problema. Problemas de "pequenas" dimensões podem ser resolvidos exactamente em tempo "aceitável". Os métodos exactos são ainda importantes na análise dos aproximativos.

O problema do caixeiro viajante múltiplo não representa por si um problema de muita aplicabilidade prática, sendo contudo importante por ser generalizável a uma vasta gama de problemas com "bastante" aplicabilidade.

O presente trabalho encontra-se organizado da seguinte forma:

- No capítulo 2, o problema é definido e formulado como um problema de programação inteira. São então resumidos alguns resultados teóricos que constituem a base de suporte para os diversos métodos de resolução.

- O método heurístico implementado bem como o método utilizado no cálculo de minorantes, apresentados nos capítulos 4 e 5 repectivamente, pressupõem ambos uma árvore com restrições de grau, cuja determinação se detalha no capítulo 3.

- O capítulo 6 documenta o método exacto utilizado na determinação da solução óptima do problema.

- As diversas técnicas implementadas permitiram a obtenção dos resultados coligidos no capítulo 7, sobre os quais se efectua uma breve análise.

- O capítulo 8 foi reservado para descrever as conclusões a retirar da observação dos resultados computacionais obtidos e discutir algumas linhas de investigação futuras no âmbito do problema em análise.

No início de cada capítulo é feita uma breve exposição dos diferentes métodos conhecidos relacionados com o tema em causa.

2. O Problema do Caixeiro Viajante Múltiplo

2.1. Introdução

O Problema do Caixeiro Viajante Múltiplo ("The Multiple Traveling Salesman Problem" : *m - TSP*) consiste em determinar o conjunto de percursos a serem efectuados por **m** caixeiros, num dado conjunto de **n** cidades (para as quais se conhece a distância entre quaisquer duas cidades). Todos os caixeiros iniciam a viagem numa cidade, préviamente fixada - cidade inicial ou depósito - onde têm que regressar no final da viagem. Cada cidade é visitada uma única vez por um só caixeiro e pretende-se que a distância total percorrida seja mínima.

O *m-TSP*, não representando por si um problema de muita aplicabilidade prática, é importante por ser generalizável a uma vasta gama de problemas com "bastante" aplicabilidade. Estes, resultam do *m-TSP* por introdução de restrições adicionais, como por exemplo:

- existência de um limite superior para a distância total percorrida por cada caixeiro;
- existência de um número máximo de cidades que cada caixeiro pode visitar;
- restrições de carga máxima e de procura nos vértices;

Problemas de distribuição ou recolha de produtos constituem casos práticos de introdução deste tipo de restrições no *m-TSP*, em que cada caixeiro representa um veículo de capacidade limitada.

O número de caixeiros, **m**, pode ser fixo ou uma variável limitada, caso em que se pretenderia ainda saber qual o número mínimo de caixeiros necessários de forma a minimizar os custos respeitando as restrições. Este trabalho incide apenas sob o caso "mais simples" em que **m** é fixo.

São apresentadas na secção seguinte as principais definições e a notação utilizada ao longo do texto, ao que se segue a formulação matemática do problema. Por fim são resumidos alguns resultados teóricos que servem de base a diferentes formas de estudo deste problema.

2.2. Formulação Matemática do Problema

A primeira formulação do *m-TSP* surgiu em 1960, tendo sido apresentada por *Miller et al* [38]. Estes autores definem e formulam, pela primeira vez, o problema seguinte:

- Partindo da cidade inicial, um caixeiro viajante tem que visitar cada uma de **n** cidades exactamente uma vez. Durante a viagem é exigido que visite a cidade inicial **m** vezes (onde **m** é variável), incluindo a viagem de regresso, e ainda que cada circuito não contenha mais de **p** cidades (entendendo-se por circuito uma sucessão de cidades visitadas sem passagem pela inicial). Pretende-se determinar um percurso que minimize a distância total percorrida.

De acordo com a definição apresentada, este problema não é mais que um *m-TSP* com uma restrição adicional, em que se exige que cada caixeiro não visite mais que um número fixo de cidades.

Neste ponto formula-se o problema de acordo com a formulação proposta por *Gavish & Srikanth* [21], sendo préviamente apresentadas algumas definições e notação básica.

Definições e Notação Utilizada

O Problema, tal como foi descrito, pode ser representado através de um grafo G = (V,A) onde:

V = { **1, 2 ,..., n** } é o conjunto dos vértices do grafo representando as **n** cidades do problema;

e,

A o conjunto das arestas ou arcos do grafo, representando as ligações entre as cidades.

A cada aresta, ou arco, encontra-se associado um custo (interessa por vezes em vez de custo considerar a distância, o valor, etc.) que se representará por:

c_{ij} - custo de um caixeiro viajar da cidade $i \in V$ para a cidade $j \in V$;

$C = [\, c_{ij} \,]_{\substack{i = 1,\ldots, n \\ j = 1,\ldots, n}}$ matriz dos custos.

A matriz **C** diz-se:

- <u>simétrica</u> se $\forall\ i,j \in V\ c_{ij} = c_{ji}$, neste caso dir-se-á que o problema é simétrico, ou que se está perante o caso simétrico do problema;

  c.c. o problema diz-se assimétrico (*);

- <u>euclideana</u> se satisfaz a norma euclideana;

Se **C** fôr simétrica o conjunto **A** designa-se por conjunto de <u>arestas</u> e **G** representa um <u>grafo não orientado</u>, caso contrário **A** representa o conjunto de <u>arcos</u> do grafo e **G** um <u>grafo orientado</u>.

Em grafos não orientados define-se <u>grau de um vértice v</u> como sendo o número de arestas incidentes em **v** e representa-se por **grau(v)**.

(*) <u>Nota</u> : Neste trabalho será apenas focado o caso simétrico e em que é válida a desigualdade triangular.

Seja :

1 - cidade de partida e de chegada, por vezes também designada por cidade inicial e final ou, simplesmente, por depósito;

$\mathbf{S = V \setminus \{1\}}$ - conjunto dos vértices do grafo com excepção do inicial;

$\mathbf{m}$ = nº de caixeiros viajantes.

Define-se :

- cadeia de comprimento $\mathbf{k}$ em $\mathbf{G = (V,A)}$ como sendo uma sequência de arestas:

$$\{ a_{i_1} , a_{i_2} , \ldots , a_{i_k} \} \qquad (a)$$

em que a_{i_r}, $2 \leq r \leq k-1$ tem um vértice comum com $a_{i_{r-1}}$, e o outro com $a_{i_{r+1}}$.

- Os vértices de a_{i_1} e a_{i_k} que não são comuns a vértices de a_{i_2} e $a_{i_{k-1}}$, respectivamente, dizem-se extremos da cadeia;

- Um caminho de comprimento $\mathbf{k}$ é uma sequência de arcos (a) onde o vértice final de $a_{i_{r-1}}$ é igual ao inicial de a_{i_r}, $1 < r \leq k$;

- Um ciclo (circuito) de comprimento $\mathbf{k}$, é uma cadeia (caminho) de comprimento $\mathbf{k}$ em que os extremos coincidem;

- Um grafo diz-se conexo se para qualquer par de vértices existir uma cadeia que os una;

- Um subgrafo de $G = (V, A)$ é um grafo $G_s = (X_s, A_s)$, onde $X_s \subseteq X$ e A_s é o conjunto de arestas de G que ligam vértices de X_s, ou seja:

$$A_s = \{ (i,j) \in A : i \in X_s, j \in X_s \};$$

- Um grafo parcial de G é um grafo $G_p = (V, A_p)$ onde $A_p \subseteq A$;

Designar-se-á por subcircuito imediato um circuito de comprimento dois, i.é.: $\{(1,i), (i,1)\}$, $i \in S$. Tal subcircuito representa o percurso de um caixeiro, que saindo da cidade inicial **1**, apenas visita a cidade **i**, regressando de seguida à cidade de partida, **1**.

Num problema em que $m \ll n$ (se $n = m$ o problema estava resolvido, e se $n < m$ é impossível) é óbvio que qualquer solução admissível do *m-TSP* tem no máximo **m-1** subcircuitos imediatos, pois se **m-1** caixeiros apenas visitam uma cidade as restantes **n-(m-1)-1** têm que ser visitadas pelo caixeiro ainda não considerado (Fig. 2.1).

Ex.: $n = 6$ (nº de cidades); $m = 3$ (nº de caixeiros);

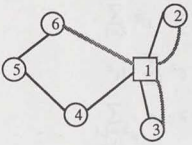


Fig. 2.1

Este facto é utilizado na determinação de minorantes, como se verá no capítulo 5.

Formulação

Definindo as variáveis binárias :

$$x_{ij} = \begin{cases} \mathbf{1} \text{ se um caixeiro viaja directamente da cidade } i \in V \text{ para a cidade } j \in V \\ \mathbf{0} \text{ caso contrário (c.c.)} \end{cases}$$

Considerando o caso simétrico e **m** fixo, o *m-TSP* pode ser formulado ([21]) como um problema de programação inteira, e consiste em :

- Determinar o valor das variáveis binárias x_{ij} que satisfazem :

(P)

$$z^* = \text{Min} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij}\, x_{ij} + \sum_{i \in S} c_{i1}\, x_{i1} \qquad (1)$$

s.a:

$$\sum_{j \in S} x_{1j} = m \qquad (2)$$

$$\sum_{i \in S} x_{i1} = m \qquad (3)$$

$$x_{j1} + \sum_{i<j} x_{ij} + \sum_{i>j} x_{ji} = 2 \quad \forall\, j \in S \qquad (4)$$

$$\sum_{\substack{i,j \in S_k \\ i<j}} x_{ij} \leq |S_k| - 1 \quad \forall\, S_k \subseteq S,\ S_k \neq \emptyset \qquad (5)$$

$$x_{ij} = 0,1 \qquad 1 \leq i < j \leq n \qquad (6)$$

$$x_{i1} = 0,1 \qquad \forall\, i \in S \qquad (7)$$

$(\text{Se } i,j \in S \quad x_{ij} = x_{ji})$

Onde as restrições :

(2) garante que da cidade (vértice) inicial, **1**, partem exactamente **m** caixeiros (arestas);

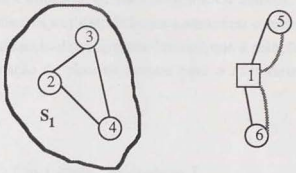
(3) garante que os **m** caixeiros regressam à cidade de partida (existem **m** arestas incidentes em **1** representando as arestas de retorno dos caixeiros);

(4) garantem que em cada vértice de **S** incidem apenas duas arestas, ou seja, o grau de cada vértice é dois. Assim, cada cidade, com excepção da inicial, é visitada uma e uma só vez;

(5) impedem a formação de subcircuitos nos subconjuntos de **S**, i.é. para qualquer $S_k \subseteq S$ existe sempre uma aresta que une um vértice de S_k com um vértice de $S \backslash S_k$, pois o número de arestas que ligam dois vértices de S_k nunca é superior a $|S_k| - 1$.

Caso estas restrições não fossem consideradas poderia ter-se, para um dos caixeiros, uma solução como a exemplificada na Fig. 2.2, não representando uma solução admissível para o *m-TSP*.

Ex.: $n = 6$; $m = 2$



$S_1 \subset S = V \setminus \{1\}$; $|S_1| = 3$; $x_{2,3} + x_{2,4} + x_{3,4} = 3$

Fig. 2.2

A função objectivo (1) garante a minimização da distância total, ou seja da soma das distâncias percorridas por cada um dos **m** caixeiros.

2.3. Outras Abordagens

Sendo uma generalização do problema do Caixeiro Viajante (*TSP* - "Traveling Salesman Problem"), não é de estranhar que os métodos existentes para a determinação de soluções aproximadas e exactas para o *m-TSP* surjam como generalizações dos diversos existentes para o *TSP*. A heurística utilizada é, como se verá, um exemplo destas técnicas.

Um estudo alternativo deste problema, pode ser feito por transformação do *m-TSP* num *TSP*. Alguns autores apresentam transformações admissíveis, que resultam da expansão do grafo que representa o *m-TSP*.

Sendo no início de cada capítulo resumidos alguns dos diferentes métodos aplicáveis ao *m-TSP*, faz-se, neste ponto, uma breve descrição de certos resultados teóricos julgados relevantes.

Bellmore & Hong [4], consideram o *m-TSP* assimétrico em que **m** é variável e em que existem custos associados à utilização de cada caixeiro. Os autores, pretendem ainda determinar o número de caixeiros a utilizar de forma a minimizar o custo total das viagens adicionado ao custo de utilização dos caixeiros. Provam que a solução deste problema coincide com a determinação do circuito óptimo para o *TSP* assimétrico no grafo expandido:

$G' = (V', A')$

onde:

$V' = V \cup \{$ **m-1** cópias do depósito $\}$

$= \{1, 2, ..., n, -1, -2, ..., -(m-1)\}$

A' contém:

todos os arcos de **A**;

(-i, j) para todos os vértices adicionais **-i**, com **j = 2,...,n** se em **A** existe o arco **(1,j)**;

(j, -i) para todos os vértices adicionais **-i**, com **j = 2,...,n** se em **A** existe o arco **(j,1)**;

(-i, -(i-1)) **i = 1, 2, ..., (m-1)**

Sendo, no grafo inicial **G**:

$d_{i,j}$ - a distância da cidade **i** à cidade **j**, **i, j = 2, 3, ..., n**;

c_k - o custo associado à utilização do **k**-ésimo caixeiro;

A distância em **G'**, é definida por:

$d'_{i,j} = d_{i,j}$. **i , j = 2, 3, ..., n;**

$d'_{-i,j} = d_{1,j} + \frac{1}{2} c_i$ **i = 1, 2, 3, ..., (m-1), j = 2, 3, ..., n;**

$d'_{j,-i} = d_{j,1} + \frac{1}{2} c_i$ **i = 1, 2, 3, ..., (m-1), j = 2, 3, ..., n;**

$d'_{-i,-(i-1)} = \frac{1}{2} c_{i-1} - \frac{1}{2} c_i$ **i = 1, 2, 3, ..., (m-1).**

Assim, um *m-TSP* assimétrico, com **n** cidades, pode ser resolvido, resolvendo um *TSP* assimétrico com **n-1+m-1** cidades.

Orloff [40] apresenta também uma transformação do *m-TSP* num *TSP* com base numa expansão do grafo original, para problemas assimétricos e uma outra para o caso simétrico. *Lenstra & Rinnooy Kan* [35] afirmam haver casos em que esta transformação não é correcta e demonstram a validade de uma nova que apresentam.

Hong & Padberg [29] transformam o *m-TSP* simétrico com **n** cidades e **m** caixeiros, em que consideram custos associados à utilização de cada caixeiro, num *TSP* simétrico com **n-1+m+4** cidades. *Discenza* [14] prova a equivalência entre a formulação deste *TSP* e uma nova em que retira dois dos vértices extra. Mostra ainda, que estas formulações são equivalentes a uma terceira em que os quatro vértices extra são substituídos por um par de restrições simples.

Mais tarde, *Rao* [42] prova que o resultado de *Bellmore & Hong* é generalizável ao caso simétrico.

Jonker & Volgenant [32], para o caso simétrico, demonstram ser possível diminuir a degenerescência originada pelas **m-1** cópias do depósito, quando $\mathbf{m > 2}$. Segundo afirmam, o maior problema desta transformação, a qual designam por transformação original, está no facto de à solução óptima do *m-TSP* corresponderem **m!** soluções óptimas do *TSP*, dadas as possíveis ordens para os depósitos fictícios. Tal facto dificulta a resolução de problemas quando $\mathbf{m > 4}$. Provando que para o *TSP*, transformação original de um *m-TSP* simétrico, existe uma solução óptima que não contém um determinado tipo de arestas - arestas removíveis - conseguem diminuir a degenerescência do *TSP*. Obtêm assim, problemas cuja resolução tem grau de dificuldade comparável ao dos *TSP's*.

Com base nestas transformações, há autores que resolvem o *m-TSP*, por um dos diversos métodos existentes para o *TSP*, como se frizará ao longo do presente trabalho.

Os métodos para a determinação de minorantes baseiam-se frequentemente em alterações na matriz de distâncias. *Berenguer* [5] mostra que as únicas transformações lineares admissíveis, ou seja, aquelas que preservam a ordem total do conjunto de soluções admissíveis de acordo com o critério de valor, para a matriz de distâncias, **C**, de um *m-TSP*, são as que se obtêm por adição de constantes θ_i e θ_j à linha **i** e à coluna **j** de $\lambda\mathbf{C}$ (com $\mathbf{i, j = 1,..., n}$; λ constante). *Lenstra & Rinnooy Kan* [36] apresentam este mesmo resultado de forma mais explícita.

3. Determinação de uma Árvore de Suporte com Restrições de Grau num Vértice, de Custo Mínimo

3.1 Introdução

O problema da determinação da árvore de suporte de custo mínimo num grafo G = (V,A) (*SST* - "Shortest Spanning Tree"), pode traduzir-se pela procura de um grafo parcial conexo sem ciclos, que contenha todos os vértices de G, de custo mínimo (*).

Planeamento de redes de transportes, sistemas de comunicação, instalação de redes de condutas de água e de electricidade são exemplos de aplicação prática do problema da determinação de uma *SST*. Este constitui ainda uma ferramenta essencial a diversos métodos utilizados na determinação de limites para certos problemas, como por exemplo o *TSP*.

Existem, para a sua resolução, diversos algoritmos considerados eficientes. *Prim* [41], e independentemente *Dijkstra* [13], desenvolveram um algoritmo de complexidade $O(|V^2|)$, eficiente para grafos simétricos e completos (onde |A|=n(n-1)/2). Para grafos pouco densos (ou seja, grafos em que |A| << n(n-1)/2), *Kruskal* [33] apresenta um algoritmo de complexidade O(|A| log |V|). *Yao* [45], por sua vez, implementa de forma mais eficiente este algoritmo, reduzindo a sua complexidade para O(|A| log log |V|). O mesmo conseguem *Cheriton & Tarjan* [7] que descrevem dois algoritmos para grafos pouco densos. Estes autores apresentam ainda um algoritmo eficiente para grafos planares

(*) Nota: O custo de uma árvore de suporte é a soma dos custos associados às arestas que a compõem.

(O(|V|)) e outro para grafos densos (O(|A|)), referindo no seu trabalho *Kerschenbaum & Van Slyke* a propósito da implementação do algoritmo de *Prim* de forma mais eficiente.

Hu [30], estuda um problema de comunicações telefónicas, envolvendo uma *SST* um pouco diferente da usual. Supõe existir um número fixo de chamadas telefónicas entre qualquer par de cidades, $\mathbf{r_{ij}}$, e define distância entre as cidades **i** e **j** como sendo o produto de $\mathbf{r_{ij}}$ com a soma das distâncias associadas às arestas no caminho único existente entre **i** e **j**, na árvore de suporte. O problema resume-se então a determinar uma árvore de suporte cujo custo total seja minimo, sendo este dado pela soma das distâncias entre os $\binom{n}{2}$ par de cidades.

Spira & Pan [43] abordam o problema da determinação de uma *SST* num grafo com mais um vértice (ou menos um; ou em que foi alterado o custo de uma aresta) que outro grafo para o qual dispõem de uma *SST*.

Com bastante interesse prático, surgem ainda diversas generalizações da *SST* que consistem na introdução de restrições adicionais - restrições de grau nos vértices e/ou de capacidades nas arestas - passíveis contudo de originar problemas de "difícil" resolução. *Garey & Johnson* [19], apresentam diversos casos relacionados com a *SST* onde enumeram alguns problemas de "difícil resolução".

Porém, as situações em que se pretende encontrar uma árvore de suporte, na qual seja fixo o grau de um só vértice, de menor custo possível, traduzem problemas de "fácil" resolução, existindo para o efeito algoritmos que os resolvem em tempo de execução polinomial. A primeira abordagem deste problema foi feita por *Glover & Klingman* [23], tendo posteriormente *Gabow* [18] apresentado e demonstrado a convergência de um algoritmo mais eficiente para grafos pouco densos.

No presente capítulo descreve-se o algoritmo de *Gabow* para a determinação de uma árvore de suporte com restrições de grau no vértice **1**, de menor custo possível.

Como se verá adiante, a heurística utilizada no cálculo de um majorante para o *m-TSP*, requere a determinação de uma árvore de suporte onde o vértice inicial, **1**, tenha grau **2m**, de menor custo possível. O método utilizado na determinação de minorantes necessita do cálculo de uma árvore análoga, em que o vértice inicial, **1**, tenha grau **m**. Assim, o algoritmo de *Gabow* será apresentado para o problema da determinação de uma árvore de suporte, onde **grau(1) = b**, de custo mínimo, que se passará a designar por *b-SST*.

O processo inicia-se com uma árvore de suporte, **T**, que contenha o conjunto de arestas incidentes em **1** no grafo inicial, de menor custo possível. As arestas de **T** incidentes em **1** vão sendo trocadas por arestas que não incidam em **1**, até ser **b** o grau do vértice **1**. A escolha da árvore inicial bem como as trocas de arestas são feitas com base nas propriedades que se enunciam e demonstram ([18]) no parágrafo seguinte. Seguidamente descreve-se o algoritmo, sendo por fim discutidas a sua convergência e complexidade.

3.2. Propriedades

As propriedades ([18]) que se apresentam neste parágrafo, têm por objectivo demonstrar a validade do algoritmo utilizado na determinação de uma árvore de suporte, onde o vértice **1** tenha grau **b**, de menor custo possível (*b-SST*).

Começando por demonstrar ser possível a obtenção de uma *(k-1)-SST* a partir de uma *k-SST*, descreve-se então uma forma de determinação de uma *r-SST*, em que **r** é o grau do vértice **1** no grafo inicial. A partir desta *r-SST*, e com base nas propriedades enunciadas, obtem-se uma *(r-1)-SST*. O processo vai-se repetindo, ou seja vão-se obtendo sucessivas árvores em que o grau de **1** vai diminuindo de uma unidade até atingir o valor de **b** (*b-SST*).

No que se segue, define-se:

* componentes conexas de um grafo $G = (V, A)$ como sendo os subgrafos gerados pelas classes de equivalência definidas pela relação (Fig 3.1):
  $i \mathcal{R} j$ sse $i = j$ ou existe uma cadeia ligando i e j, com $i,j \in V$;

* floresta de suporte de um grafo **G** como sendo o grafo definido pelo conjunto das árvores de suporte para cada componente conexa de **G** (*) (Fig. 3.2).

(*) Nota: Se o grafo fôr conexo só existe uma componente conexa e, consequentemente, a floresta de suporte será formada por uma só árvore de suporte.

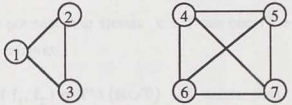
e considera-se:

R o conjunto das arestas de **G = (V,A)** incidentes em **1**;

$\mathfrak{T}_k$ o conjunto de todas as árvores de suporte, **T**, tais que **grau(1) = k**

$$\therefore\ T \in \mathfrak{T}_k \quad \text{se} \quad k = |\, R \cap T\,| \ (*)$$

Ex.: Seja o grafo seguinte:



(a) Fig. 3.1. (b)

(a) e (b) representam as duas componentes conexas do grafo

Uma floresta de suporte é dada por:

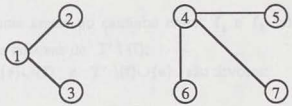


Fig. 3.2.

Assim, o problema que se pretende resolver consiste em determinar uma árvore em $\mathfrak{T}_b$ de custo menor possível ([18]).

Teor. 1: Seja $T \in \mathfrak{T}_k$ de custo mínimo. Então:

(a) Se $\mathfrak{T}_{k-1} \neq \emptyset$

existem arestas $e \in R \cap T$ e $f \notin R \cup T$ tais que $T \setminus \{e\} \cup \{f\}$ é uma árvore em $\mathfrak{T}_{k-1}$ de custo mínimo;

(*) Nota: Com o objectivo de simplificar a notação, **T** tanto denota uma árvore como as suas arestas.

(b) Se $\mathfrak{T}_{k+1} \neq \emptyset$
existem arestas $e \in T \backslash R$ e $f \in R \backslash T$ tais que $T \backslash \{e\} \cup \{f\}$ é uma árvore em $\mathfrak{T}_{k+1}$ de custo mínimo.

Dem.:

(a) Seja $T' \in \mathfrak{T}_{k-1}$ de custo mínimo, contendo tantas arestas de T quantas possíveis;

Começa-se por encontrar arestas e e f tais que $\hat{T} = T \backslash \{e\} \cup \{f\} \in \mathfrak{T}_{k-1}$ e tenha custo mínimo;

Seja $f = (f_1, f_2) \in T' \setminus (R \cup T)$ (f existe, pois $|T' \backslash R| > |T \backslash R|$)

Como T é uma árvore de suporte, existe caminho entre qualquer par de vértices, em T $\Rightarrow$ existe caminho entre f_1 e f_2, em T;
$f \in T' \Rightarrow T' \setminus \{f\}$ representa duas componentes conexas;

Seja e uma aresta no caminho entre f_1 e f_2 em T que liga as duas componentes conexas de $T' \setminus \{f\}$;
Logo, $T \backslash \{e\} \cup \{f\}$ e $T' \setminus \{f\} \cup \{e\}$ são árvores;

Demonstre-se agora que $e \in R$:
Para tal, suponha-se que $e \notin R$, então:

$$\left.\begin{array}{l} (T \backslash \{e\} \cup \{f\}) \in \mathfrak{T}_k \Rightarrow c_e \leq c_f \\ (T' \backslash \{f\} \cup \{e\}) \in \mathfrak{T}_{k-1} \Rightarrow c_f \leq c_e \end{array}\right\} \Rightarrow c_e = c_f$$

Logo, $(T' \backslash \{f\} \cup \{e\}) \in \mathfrak{T}_{k-1}$ tem custo mínimo e contém mais arestas de T do que T', o que contradiz a definição de T'

$\therefore$ $e \in R$, como por definição $e \in T \Rightarrow e \in R \cap T$

Resta provar que $\hat{T} = T \backslash \{e\} \cup \{f\}$ tem custo mínimo em $\mathfrak{T}_{k-1}$

$$\hat{T} \in \mathfrak{T}_{k-1} \Rightarrow c(\hat{T}) \geq c(T') \qquad (1)$$

$(T'\backslash\{f\}\cup\{e\}) \in \mathfrak{T}_k \Rightarrow c(T'\backslash\{f\}\cup\{e\}) \geq c(T) \Rightarrow$

$\Rightarrow c(T') - c_f + c_e \geq c(T) \Rightarrow c(T') \geq c(T) + c_f - c_e \Rightarrow$

$\Rightarrow c(T') \geq c(\hat{T})$ (2)

(1) e (2) $\Rightarrow c(\hat{T}) = c(T')$

$\therefore \hat{T} = T\backslash\{e\}\cup\{f\}$ é uma árvore em $\mathfrak{T}_{k-1}$ de custo mínimo;

(b) demonstra-se de forma análoga. (c.q.d.)

Cor. 1: Seja $T \in \mathfrak{T}_k$ de custo mínimo.

Se $e \in R\cap T$ e $f \notin R\cup T$ tais que:

$T\backslash\{e\}\cup\{f\}$ é uma árvore

e

$c_f - c_e$ é mínimo entre tais arestas

então, $T\backslash\{e\}\cup\{f\} \in \mathfrak{T}_{k-1}$ e tem custo mínimo.

Dem.: $T \in \mathfrak{T}_k$; $e \in R\cap T$; $f \notin R\cup T$; $T\backslash\{e\}\cup\{f\}$ é uma árvore $\Rightarrow$

$\Rightarrow \hat{T} = T\backslash\{e\}\cup\{f\} \in \mathfrak{T}_{k-1}$

Seja, $T' \in \mathfrak{T}_{k-1}$ de custo mínimo $\Rightarrow c(T') \leq c(\hat{T})$ (1)

Pelo teor.1, existem arestas:

$e_1 \in R\cap T$, $f_1 \notin R\cup T$ tais que $c(T') = c\left(T\backslash\{e_1\}\cup\{f_1\}\right) \Rightarrow$

$\Rightarrow c(T') = c(T) - c_{e_1} + c_{f_1} \geq c(T) - c_e + c_f$ (2)

(1) e (2) $\Rightarrow c(T') = c(\hat{T})$

$\therefore \hat{T} = T\backslash\{e\}\cup\{f\} \in \mathfrak{T}_{k-1}$ e tem custo mínimo (c.q.d.)

O teor.1 e o cor.1 indicam como se pode encontrar uma árvore de suporte onde o grau do vértice **1** seja **k-1**, de custo mínimo, a partir de uma árvore de suporte onde **1** tenha grau **k**, de custo mínimo. Torna-se assim evidente que se fôr resolvido o problema da determinação de uma árvore de suporte, onde o vértice **1** tenha grau igual ao que tinha no grafo inicial, $\mathbf{r} = |R|$, de menor custo possível se pode, por sucessivas aplicações destes

resultados determinar uma *b-SST*, desde que $b \leq r = |R|$ (pois, caso contrário o problema é impossível).

Cor. 2: Seja **U** uma floresta de suporte de custo mínimo em $G_1 = (S, A_1)$ (onde $S = V \setminus \{1\}$ e A_1 representa o conjunto de arestas em **G** que ligam vértices de **S**) e U_A o conjunto de arestas de **U**.

Então,

$G_2 = (V, U_A \cup R)$ contém uma árvore em $\mathfrak{T}_r$ de custo mínimo, se $\mathfrak{T}_r \neq \emptyset$.

Dem.:

Qualquer árvore de suporte de **G** contém uma aresta de **1** para cada árvore de **U**;

Seja **k** o número de árvores em **U**, $k \leq r$ (**k** é o menor valor que o grau de **1** pode tomar);

Uma árvore $T' \in \mathfrak{T}_k$ de custo mínimo, pode obter-se unindo as **k** arestas de menor custo ($R_k \subset R$) que ligam **1** a cada árvore de **U**;

$\therefore$ $T' = (V, U_A \cup R_k) \in \mathfrak{T}_k$ e tem custo mínimo;

Utilizando o teor.1(b), pode encontrar-se com base em **T'** uma árvore em $\mathfrak{T}_{k+1}$ de custo mínimo, se $\mathfrak{T}_{k+1} \neq \emptyset$, contendo apenas arestas de $U_A \cup R$;

Repetindo o processo determina-se em $\mathfrak{T}_r$ a árvore pretendida, se $\mathfrak{T}_r \neq \emptyset$;

$\therefore$ se $\mathfrak{T}_r \neq \emptyset$, G_2 contém uma árvore em $\mathfrak{T}_r$ de custo mínimo. (c.q.d.)

Obs.: Veja-se em que casos $\mathfrak{T}_r \neq \emptyset$:

Seja $T' = (V, U_A \cup R_k)$;

Se $|R| \geq k$ e $\mathfrak{T}_k \neq \emptyset$ existem **r-k** arestas incidentes em **1** pertencentes a $R \setminus R_k$;

Se forem juntas a **T'** estas arestas e retiradas outras **r-k** de forma a se ter uma árvore (logo, serão arestas de U_A), obtém-se uma árvore em $\mathfrak{T}_r$;

$\therefore$ $\mathfrak{T}_r \neq \emptyset$;

Ex: Seja

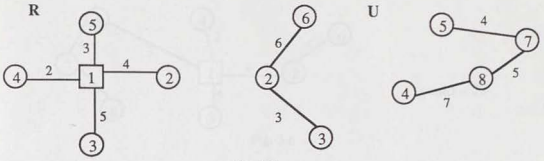
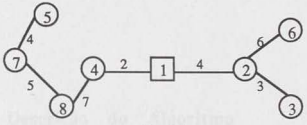


Fig. 3.3

$k = 2$ (nº de árvores de U) $\quad R_2 = \{(1,4), (1,2)\}$

∴ A árvore $T' \in \mathfrak{T}_2$ de custo mínimo é:



custo = 31

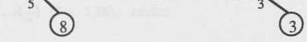
Fig. 3.4

Introduzindo a aresta $(1,3) \in R \backslash R_2$ em T' tem que ser retirada $(2,3) \in U_A$, sendo: $c_{1,3} - c_{2,3} = 2$;

Introduzindo a aresta $(1,5) \in R \backslash R_2$ em T' é retirada (5,7) ou (7,8) ou (4,8), sendo: $c_{1,5} - c_{5,7} = -1$; $\quad c_{1,5} - c_{7,8} = -2$; $\quad c_{1,5} - c_{4,8} = -4$;

∴ substitui-se (1,5) por $(4,8) \in U_A$, pois representa a troca de arestas que provoca menor "aumento" no custo total da árvore.

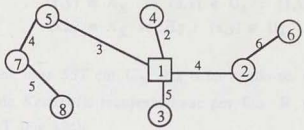
∴ Uma árvore em $\mathfrak{T}_3$ de custo menor possível é:



custo = 27

Fig. 3.5

e em $\mathfrak{T}_4$:



custo = 29

Fig. 3.6

Nota: Se G_1 fôr conexo, **U** representa uma *SST*, sendo **k = 1**, mantendo-se a validade do corolário 2.

3.3. Descrição do Algoritmo

No algoritmo implementado começa-se o processo com uma árvore de suporte, **T**, que contenha o conjunto **R** de arestas incidentes no vértice **1** (ou seja com uma árvore em que **1** tenha grau (>**b**) igual ao que tinha no grafo inicial, **G**), de menor custo possível.

No parágrafo anterior, cor.2, provou-se que, sendo **U** uma floresta de suporte de custo mínimo em $G_1 = (S, A_1)$, o grafo $G_2 = (V, U_A \cup R)$ contém uma árvore de suporte, onde **grau(1)=|R|** de custo mínimo (*/R/-SST*).

Assim, para se obter uma */R/-SST* , **T**, começa-se por encontrar uma floresta de custo mínimo, **U**, em G_1. Tal foi feito com base no algoritmo de *Kruskal* ([44]) para a determinação de uma árvore de suporte de custo mínimo (*SST*), em que se G_1 fôr desconexo se tem como resultado uma floresta de suporte de custo mínimo. Dada **U**, considera-se o grafo G_2 (Fig. 3.7a). Encolhe-se **R** a um ponto, s, resultando um grafo $G_X = (X, A_X)$ (Fig. 3.7b), onde:

X é o conjunto formado pelo vértice **s** e todos os vértices não adjacentes a **1** em **G**, $|X| = n - grau(1) + 1$;

e

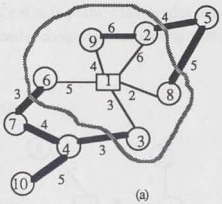
A_X é tal que:

$(x,y) \in A_X$ se $(x,y) \in U_A$; $(1,x) \notin R$ e $(1,y) \notin R$;

$(x,s) \in A_X$ se $\exists y$: $(x,y) \in U_A$; $(1,y) \in R$ e $(1,x) \notin R$.

Calcula-se uma *SST* em G_X (Fig. 3.7c) (tendo-se, mais uma vez, recorrido ao algoritmo de *Kruskal*), recuperando-se por fim **R**, de forma a obter a */R/-SST* pretendida, **T** (Fig. 3.7d).

Ex: Seja $n = 10$ e G_2:

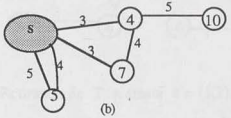


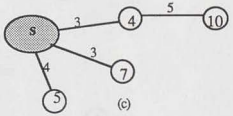
onde:

/ representam arestas de **R**

representam arestas de **U**

G_X





T

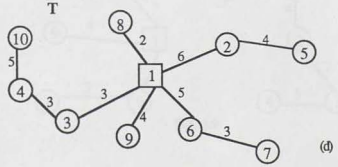


Fig. 3.7

Após se ter encontrado a árvore de suporte inicial, **T**, na qual **grau(1)** = |R| (>**b**) e determinada **U**, começa-se a fase de trocas entre as arestas de **U\T** e as de **T** incidentes em **1**. Esta fase termina quando **grau(1)** = **b**.

Com o objectivo de escolher a melhor troca possível em cada caso, criam-se filas de prioridades para cada aresta de **T∩R** de acordo com o seguinte procedimento:

A aresta **e ∈ T∩R** pode ser substituída por **f ∈ U** se **f** unir as duas componentes conexas de **T\{e}**. A fila de prioridades **F(e)** guarda todas as arestas do tipo de **f**. A prioridade de cada aresta, representando a alteração no custo total da árvore, **T**, se a aresta **e** fôr trocada por **f**, é dada por $c_f - c_e$ (sendo c_e constante em **F(e)**).

Assim, em cada passo, são trocadas as arestas de menor valor de diferença de custos, sendo referidas como sendo as arestas de maior prioridade.

Ex.: Seja

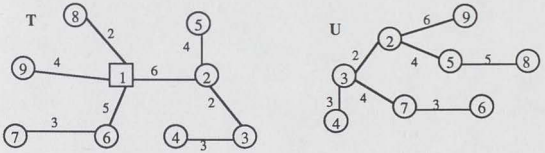


Fig. 3.8

Retirando de **T** a aresta **e = (1,2)**, resulta:

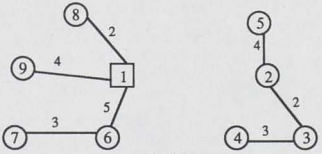


Fig. 3.9

as arestas $f \in U$ que podem substituir $e = (1,2)$ são arestas que liguem um dos vértices do conjunto $B = \{2,3,4,5\}$ com um vértice de $S \backslash B$. Assim, a fila de prioridades de e vem:

$$F(1,2) = \begin{bmatrix} (2,9) & (3,7) & (5,8) \\ 0 & -2 & -1 \end{bmatrix} \quad \begin{matrix} \text{(arestas)} \\ \text{(prioridades)} \end{matrix}$$

Sendo T uma *k-SST* e $c_{f_1} - c_{e_1} = \underset{e \in T \cap R}{\text{Min}} \{ \underset{f \in F(e)}{\text{Min}} \{ c_f - c_e \} \}$ sabe-se que (cor.1) $T \backslash \{e_1\} \cup \{f_1\}$ é uma *(k-1)-SST*.

O problema da determinação de uma *b-SST* não tem solução se o grau do vértice **1**, no grafo original, fôr menor que **b**, nem se não existirem arestas de **1** para cada uma das subárvores de **U** (Fig. 3.10). O presente trabalho incide apenas sobre problemas em grafos completos para os quais existe sempre uma *b-SST* ($b \leq n-1$). Contudo, no método utilizado para a determinação da solução óptima (cap. 6) são resolvidos, como se verá, subproblemas em que algumas das arestas do grafo inicial são fixadas com valor zero, podendo resultar um problema para o qual não exista uma *b-SST*. Tal facto, levou a que na implementação computacional se incluísse um teste à impossibilidade do problema da determinação da *b-SST*.

Ex.: Seja $b = 2$

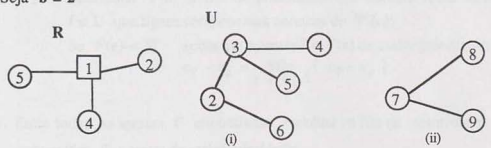


Fig. 3.10

Embora $|R| = 3 > 2$, não existem em **R** arestas para a subárvore (ii) de **U**. Logo, o grafo é desconexo não existindo nenhuma *2-SST*.

Algoritmo (I): (Determinação de uma *b-SST* em **G**)

Dados: Matriz dos custos, **C**;
nº de cidades, **n**;
valor que se pretende fixar como grau do vértice **1**, **b**;

Resultado: **T** - Vector representativo das arestas da *b-SST*;
csst - custo da *b-SST*;

1. Determinar **U** uma floresta de suporte mínima em $G_1 = (S, A_1)$;
Seja **R** o conjunto das arestas de **G** incidentes em **1**;
Fazer **grau(1)** ← |**R**|;
Se **grau(1) < b** ou se em **R** não existirem arestas de **1** para cada uma das subárvores de **U** terminar - Problema impossível;
c.c. continuar;

2. Encontrar **T** uma */R/-SST*, i.é. uma árvore de suporte que contém **R**, de menor custo possível;
Calcular o seu custo: **csst**;
Se **grau(1) = b** terminar - **T** é uma *b-SST*;
c.c. continuar;

3. Para cada aresta
e ∈ R:
* determinar **F(e)** a fila de prioridades que contém todas as arestas **f ∈ U** que ligam componentes conexas de **T\{e}**;
Se **F(e) ≠ Ø** encontrar a aresta **f' ∈ F(e)** de maior prioridade:
$$c_{f'} - c_e = \min_{f \in F(e)} \{ c_f - c_e \}$$

4. Entre todas as arestas **f'** encontradas escolher a fila de prioridades **F(e)** que contém **f''** a aresta de maior prioridade;

5. {* substituição da aresta **e** por **f''** em **T** *}
T ← **T\{e}∪{ f'' }**;
Fazer: **grau(1)** ← **grau(1) - 1**;
csst ← **csst** + $c_{f''} - c_e$;

6. Se **grau(1) > b** ir para 7;
c.c. terminar : encontrou-se a *b-SST* pretendida;

7. {* actualização das filas de prioridades *}

7_1. Retirar **f''** de **F(e)**;

7_2. Retirar **f''** de **F(e')**, onde **e'** é a outra aresta que pode ser substituída por **f''** [(i)];

7_3. Juntar **F(e)** a **F(e')** [(ii)] calculando, para cada $f \in F(e)$, $c_f - c_{e'}$, ou seja a prioridade de **f** na nova fila de prioridades **F(e')**;

7_4. Se $F(e') \neq \emptyset$ encontrar **f'** a aresta de maior prioridade em **F(e')**;
Voltar a **4**.

Observações:

(i) A aresta **e'** referida em 7_2 existe sempre, pois dada qualquer aresta **(x,y)** de **U** ela pode sempre substituir duas (e só duas) arestas incidentes em **1**, desde que possa substituir uma. Cada aresta **f''** representa sempre a ligação entre componentes conexas de **T\{1}**, podendo assim substituir as arestas que unem **1** com os dois vértices adjacentes a **1** em cada uma dessas duas componentes conexas. Considerando o exemplo da Fig.3.8, tem-se que a aresta $(3,7) \in U$ pode substituir $(1,2) \in T$ e $(1,6) \in T$.

(ii) Em 7_3, a junção de **F(e)** a **F(e')** é justificável pelo facto de as arestas que podiam substituir **e** em **T** passarem a poder substituir **e'** em $T\backslash\{e\}\cup\{f''\}$. No exemplo em análise, se **(1,2)** fôr substituída por **(3,7)**, sendo :

$$F(1,2) = \underset{\text{(prioridade)}}{\begin{bmatrix} (3,7) & (5,8) & (2,9) \\ -2 & -1 & 0 \end{bmatrix}} \qquad F(1,6) = \begin{bmatrix} (3,7) \\ -1 \end{bmatrix}$$

passaria a ter-se :

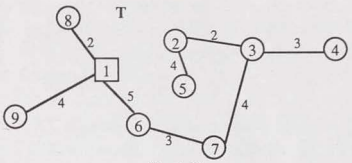


Fig. 3.11

onde, $\mathbf{F(1,2) = \emptyset}$ e $\mathbf{F(1,6)} = \begin{bmatrix} \mathbf{(5,8)} & \mathbf{(2,9)} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}$

ou seja, as arestas **(5,8)** e **(2,9)** que podiam substituir **(1,2)** passam a poder substituir **(1,6)**, muito embora a prioridade seja diferente.

Com base no que foi afirmado e nas propriedades do parágrafo anterior, torna-se óbvio concluir que o algoritmo descrito resolve o problema a que se propunha, ou seja determina uma *b-SST* num grafo **G = (V, A)**.

Utilizando, na determinação das *SST* necessárias aos passos **1** e **2**, um dos algoritmos referidos ([7], [45]) de complexidade O(|A| log log |V|), em vez do de *Kruskal* de complexidade O(|A| log |V|), *Gabow* [18] demonstra que a complexidade deste algoritmo é da ordem de O(|A| log log |V| + |V| log |V|). Porém, esta ainda pode ser melhorada se na determinação da árvore inicial, **T**, forem utilizadas técnicas semelhantes às de *Spira & Pan* [43], caso em que se obtém uma complexidade de O(|V|), em vez da actual O(|V| log|V|).

4. Determinação de um Majorante e de uma Solução Admissível

4.1. Introdução

Sendo o *m-TSP* uma extensão do problema do caixeiro viajante simples, parece natural que grande parte das heurísticas existentes para a determinação de uma solução admissível para o *m-TSP* e de um majorante para o seu valor óptimo, surjam como adaptação das existentes para o *TSP*.

Pode ainda pensar-se numa transformação do *m-TSP* num *TSP* e utilizar qualquer dos métodos existentes para o *TSP*. Contudo esta alternativa origina um aumento pouco conveniente da dimensão do problema.

Os métodos que geram soluções admissíveis para o *TSP* podem dividir-se em duas classes - métodos de construção e métodos de melhoramento. A primeira engloba heurísticas de construção de um circuito com base na matriz das distâncias, enquanto que a segunda se traduz em processos iterativos de melhoria de um circuito inicial. *Golden et al.* [24] resumem estes métodos, estudam as suas complexidades e fazem sempre que possível a análise do pior caso, i.e. indicam um majorante para a razão entre o valor da solução admissível obtida pelas heurísticas e o valor óptimo do problema.

Frederickson, Hecht & Kim [16], adaptam à determinação de uma solução admissível para o *m-TSP* dois métodos de construção para o *TSP* - o método da inserção mais próxima e o do vizinho mais próximo, que partindo de um nó inicial se diferenciam pela forma como é seleccionado o vértice seguinte a integrar no circuito. Apresentam, ainda

uma terceira heurística ("*m-Splitour*") que consiste na subdivisão de um circuito para o *TSP* em **m** subcicuitos. Este método é dos três o que apresenta melhor análise de pior caso, ou seja o que, no pior caso, obtém uma solução admissível com valor mais próximo do valor óptimo do *m-TSP*, caso seja utilizada a heurística de *Christofides* [9] para a determinação do circuito inicial para o *TSP*.

No presente trabalho, a determinação de uma solução admissível e de um majorante para o valor óptimo da função objectivo do problema foi feita de acordo com a heurística de *Frieze* [17], sendo esta por seu turno uma generalização da de *Christofides* [9]. Assim, foi adaptada a implementação computacional da heurística de *Christofides* elaborada por *Mourão* [39].

O método proposto por *Christofides* só é aplicável a problemas cujos grafos sejam completos e é o que proporciona, em casos simétricos e que verifiquem a desigualdade triangular, melhor análise de pior caso. Nestas condições, o valor da solução por ele encontrado, não é mais do que 50% superior ao valor óptimo, resultado que *Frieze* [17] demonstra manter-se válido para o *m-TSP*.

A descrição da heurística de *Frieze*, apresentada no próximo parágrafo, é antecedida pelas definições julgadas importantes. São explicados os principais passos deste método de forma a justificar a admissibilidade da solução que se obtem. Por fim é feita a comparação possível entre o valor obtido por esta heurística e o valor óptimo do problema.

4.2. Método Heurístico

Designando-se por m-percurso, **H**, um conjunto de **m** subpercursos $\mathbf{H_1},...,\mathbf{H_m}$, onde:

(i) $\mathbf{H_i}$, **i=1,...,m** forma um ciclo;

(ii) $\mathbf{H_i}$, **i=1,...,m** utiliza o vértice inicial, **1**;

(iii) para cada vértice $\mathbf{v} \in \mathbf{S} = \mathbf{V}\backslash\{\mathbf{1}\}$ existe um único ciclo que passa em **v**.

e sendo o custo total de **H** dado por:

$$c(H) = \sum_{i=1}^{m} c(H_i) \quad \text{onde} \quad c(H_i) = \sum_{e_k \in H_i} c(e_k)$$

Uma solução admissível para o *m-TSP* pode ser dada por um **m**-percurso encontrado de acordo com o método heurístico proposto por *Frieze*, que se descreve neste ponto.

Dado um caminho Euleriano no grafo, i.é. um caminho que passa uma e uma só vez por cada aresta do grafo, torna-se fácil encontrar um **m**-percurso, sendo simultaneamente garantida a sua existência. Demonstra-se que tal caminho existe em grafos Eulerianos, ou seja, grafos conexos e em que todos os vértices têm grau par ([25]).

No método que se apresenta começa-se (passos **1** e **2**) por construir, com base no grafo inicial, um grafo Euleriano, sendo para tal necessária a introdução dos seguintes conceitos:

* Dado o grafo $G_0 = (V_0, A_0)$, um subconjunto E_0 de A_0 é um emparelhamento ("*matching*") em G_0 se quaisquer duas arestas de E_0 não tiverem vértices em comum (Fig 4.1);

* Um emparelhamento E_0 diz-se perfeito se em todo o vértice de G_0 incidir uma aresta do emparelhamento (Fig 4.1), i.e., se:

$$| E_0 | = | V_0 | / 2$$

Note-se que só existem emparelhamentos perfeitos em grafos com um número par de vértices.

Obs: $| V_0 |$, que no método heurístico representa o número de vértices de grau ímpar de uma árvore de suporte, onde o grau do vértice **1** é **2m**, de menor custo possível (*2m-SST*), é um número par, pois num grafo (e, em particular numa árvore) o número de vértices de grau ímpar é par.

* O custo dum emparelhamento é igual à soma dos custos das arestas que o compõem.

Ex.: Considere-se o grafo:

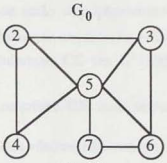


Fig. 4.1

Um emparelhamento em G_0 é, por exemplo, formado pelas arestas:

$\{(4,5),\ (6,3)\}$;

sendo um emparelhamento perfeito: $\{(4,2),\ (6,3),\ (7,5)\}$.

A solução admissível para o *m-TSP*, bem como o seu valor, foram determinados de acordo com o método heurístico que se descreve no algoritmo seguinte. Neste, com o objectivo de não sobrecarregar a notação, e dado não ser possível qualquer confusão, $\mathbf{T_0}$ tanto denota a *2m-SST* como as suas arestas.

Algoritmo (II) :

Dados: Matriz de custos, **C**; Número de cidades, **n**; Número de caixeiros, **m**;

Resultado: Solução admissível representada pelo m-percurso **H**;
Valor da solução encontrada, $\bar{z}$;

1. Determinar uma *2m-SST* em **G = (V,A)**. Seja $\mathbf{T_0}$ essa árvore (utilizou-se o Algoritmo (I) - cap. 3 - com **b = 2m**);

2. Identificar o conjunto de vértices de grau ímpar, $\mathbf{V_0}$, em $\mathbf{T_0}$;
Construir o emparelhamento perfeito de custo mínimo ([6]) $\mathbf{E_0}$ (*"1-matching"*) em $G_0 = (V_0, A_0)$, sendo $A_0 \subset A$ o conjunto de arestas que ligam os vértices de $\mathbf{V_0}$ no grafo inicial;

3. No grafo $G' = (V, T_0 \cup E_0)$, todos os vértices têm grau par se forem duplicadas as arestas que estão simultâneamente em T_0 e E_0. Logo, G' é Euleriano existindo, portanto um caminho Euleriano;
   Construir o caminho Euleriano **CE** em G' ([25]);

4. Transformar o caminho euleriano **CE** num **m**-percurso **H**, como se segue:

   Seja R_0 o conjunto dos vértices adjacentes a **1** em T_0;
   Suponha-se que **CE** segue a seguinte sequência de vértices:

   $$1 = w_1, w_2, \ldots, w_k = 1$$

   Analisa-se a sequência acima, não considerando um vértice w_i se um dos dois casos seguintes se der:
   (a) $w_i \neq 1$ e w_i já tiver sido visitado (pois cada vértice é visitado uma e uma só vez);
   (b) $w_i \in R_0$ e $1 \notin \{ w_{i-1}, w_{i+1} \}$.

5. Calcular o valor da solução, como sendo a soma dos custos das arestas pertencentes ao **m**-percurso **H**, seja $\bar{z}$ esse valor.

Cabe aqui salientar que para a realização dos percursos nunca são necessários mais de **m** caixeiros, dado que se de G' (passo **3**) fôr retirado o vértice **1**, resultam **m** subgrafos no máximo. Tal equivale a afirmar que pelo menos **m** arestas do emparelhamento unem vértices de subárvores de T_0 (resultantes de retirar **1** de T_0) diferentes, o que pode ser justificado do seguinte modo:

Em cada uma das diferentes subárvores, resultantes de T_0 sem o vértice **1**, os vértices de grau ímpar são em número par (pois, isoladamente são árvores e numa árvore o número de vértices de grau ímpar é sempre par). Seja **v** o vértice adjacente a **1** numa dessas **2m** subárvores (**2m** dado ser o grau de **1**). Juntando a aresta **(1,v)** à subárvore o número de vértices de grau ímpar passa a ser ímpar. Logo, pelo menos uma das arestas do emparelhamento tem que unir um dos vértices de grau ímpar desta subárvore com um de grau ímpar de outra diferente, dado tratar-se de um emparelhamento perfeito (no conjunto de vértices de grau ímpar) onde todos os vértices são emparelhados dois a dois.

Ex: Seja **n = 11** (nº de cidades), **m = 2** (nº de caixeiros) e T_0 dada por:

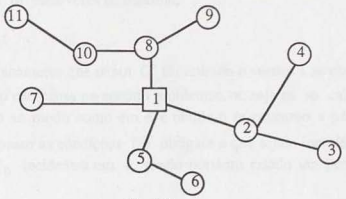


Fig. 4.2

onde, o conjunto dos vértices de grau ímpar é : $V_0 = \{2, 3, 4, 6, 7, 8, 9, 11\}$.

As 4 subárvores resultantes da eliminação do vértice **1** são:

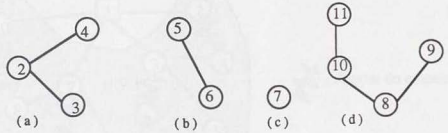


Fig. 4.3

tendo-se,

subárvore	vértices de grau ímpar	
	—	incluíndo **1**
(a)	**3, 4**	**2, 3, 4**
(b)	**5, 6**	**6**
(c)	**Ø**	**7**
(d)	**9, 11**	**8, 9, 11**

Sendo em cada um dos casos, ímpar o número de vértices de grau ímpar, pelo menos **m** das arestas do emparelhamento perfeito unem cada uma das subárvores com outra. Ou

seja pelo menos **m** das **2m** subárvores ficam ligadas por uma aresta do emparelhamento, resultando no final **m** subárvores no máximo.

Contudo, pode acontecer que se em **G'** fôr retirado o vértice **1** se obtenha apenas um subgrafo. Tal facto não causa no entanto problemas, ou seja os **m** caixeiros são todos utilizados, devido ao modo como em **4** é obtido o **m**-percurso a partir do caminho Euleriano. Neste passo as condições (b) obrigam a que sejam incluídas na solução as **2m** arestas de T_0 incidentes em **1**, sendo portanto criado um percurso para cada caixeiro (Fig. 4.4).

Ex.: **n = 11 ; m = 2**

Seja $T_0 \cup E_0$:

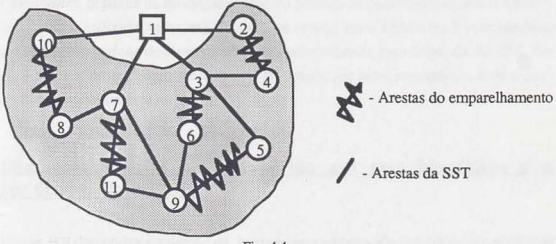


Fig. 4.4

Retirando o vértice **1** resulta o subgrafo assinalado com ponteado.

Sendo um caminho Euleriano dado por :

CE = {(1,2), (2,4), (4,3), (3,5), (5,9), (9,6), (6,3), (3,1), (1,10), (10,8), (8,7), (7,11), (11,9), (9,7), (7,1)}

e,

R_0 **= { 2, 3, 7, 10 }**

A passagem de **CE** a **H** é feita seguindo a sequência de vértices de **CE** enquanto não se tiver **4**(a) ou **4**(b). Assim, inicialmente vem:

H = {(1,2), (2,4),

como $3 \in R_0$ e $1 \notin \{(4,3),(3,5)\}$, a aresta (4,3) não é considerada para o poder ser (3,1).

O m-percurso será então dado por:

H ={ (1,2) , (2,4) , (4,5) , (5,9) , (9,6) , (6,3) , (3,1) , (1,10) , (10,8) , (8,11) , (11,7) , (7,1) }

Assim, o m-percurso contém as **2m** arestas da *2m-SST*, T_0, incidentes em **1**.

Sendo os passos **1** e **4** deste método os únicos que diferem do método heurístico de *Christofides*, a implementação computacional referida apenas aqui reflete alguma modificação.

Frieze [17] mostra que a complexidade deste algoritmo, tal como a do método heurístico de *Christofides*, depende da do emparelhamento perfeito de custo mínimo, sendo $O(|V|^3)$ a do algoritmo utilizado neste trabalho. Como se viu, nesta heurística é encontrado o emparelhamento perfeito de custo mínimo para os vértices de grau ímpar da *2m-SST*. Em geral, o cardinal deste conjunto é inferior a |S|, sendo, portanto, esta análise de pior caso muito raramente observada na prática.

Comparação entre o valor obtido por esta heurística e o valor óptimo do *m-TSP*

Frieze [17] demonstra a validade do teorema seguinte quando considerado o problema do *m-TSP* completo e cuja matriz de custos seja simétrica verificando a desigualdade triangular.

Teor.: Sendo $v(H_0)$ o valor da solução obtido por esta heurística e $v(H^*)$ o valor óptimo do problema é válida a relação:

$$v(H_0) \leq \frac{3}{2} v(H^*)$$

isto é, no pior caso, o valor encontrado pelo método heurístico de *Frieze* não é mais do que 50% superior ao valor óptimo do problema.

5. Determinação de Minorantes

5.1. Introdução

A relaxação Lagrangeana tornou-se, após o trabalho de *Held & Karp* [27], num dos métodos mais utilizados na determinação de minorantes para problemas de optimização combinatória tidos como "difíceis". Alguns destes incluem restrições "complicantes" as quais são transferidas para a função objectivo, atribuindo-se-lhes penalidades - multiplicadores de Lagrange - originando um problema, problema relaxado, de mais "fácil" resolução. Uma etapa fundamental, de acordo com estes métodos consiste na procura de solução para o problema relaxado.

No âmbito do presente trabalho, sendo o *m-TSP* um problema de minimização, o valor do problema relaxado fornece, para valores fixos dos multiplicadores, um minorante do seu valor óptimo. Este minorante pode ser melhorado com a resolução do problema dual Lagrangeano que consiste na determinação dos valores dos multiplicadores que maximizam o valor do problema relaxado. A resolução do problema dual Lagrangeano é dificultada pelo facto da sua função objectivo não ser diferenciável em todo o seu domínio. Tal pode ser contornado pelo recurso a métodos de optimização subgradiente, dado a resolução do problema relaxado fornecer sempre um subgradiente para a função objectivo do dual (*).

(*) Obs.: Um estudo aprofundado da Dualidade Lagrangeana pode ser encontrado em *Geoffrion* [22], sendo neste trabalho justificada apenas a sua aplicação ao problema em estudo.

Estes métodos para a determinação de minorantes diferenciam-se, fundamentalmente, pela formulação utilizada e pelas restrições que se relaxam.

No método proposto por *Gavish & Srikanth* [21], relaxam-se as restrições de grau nos vértices de S.

Christofides et al. [10] ao estudarem um problema de determinação de rotas para veículos (*VRP* - " Vehicle Routing Problem") descrevem um método para o cálculo de minorantes para o *m-TSP*. Utilizam a mesma relaxação que *Gavish & Srikanth*, embora com base numa formalização diferente por considerarem variável o número de caixeiros viajantes. Sendo o *VRP* um *m-TSP* com restrições adicionais, é óbvio que um minorante para o *m-TSP* representa ainda um minorante para o *VRP*. *Christofides et al.* concluem que este método não dá bons resultados para o *VRP*. Nos resultados apresentados os minorantes que obtêm para o *m-TSP* são melhorados, de forma a poderem proporcionar melhores resultados para o *VRP*, não sendo assim possível a comparação de resultados entre este método e o aqui estudado.

Ali & Kennington [1] abordam o caso assimétrico, relaxando as restrições de grau em todos os vértices do grafo, generalizando ao *m-TSP* o método proposto por *Held & Karp* [26, 27] e por *Bazaraa & Goode* [2] para o *TSP*.

Jonker & Volgenant [32], com base na transformação do *m-TSP* num *TSP* anteriormente referida (cap. 2), e no método proposto por *Held & Karp*, relaxam as restrições de grau e determinam os minorantes utilizando um método de subgradiente cuja fórmula de actualização de multiplicadores é diferente da habitualmente utilizada ([31]).

Este capítulo engloba a formulação do problema relaxado, com base na relaxação Lagrangeana utilizada ([21]) e do problema dual Lagrangeano. Segue-se a descrição do algoritmo implementado para o cálculo do valor do problema relaxado e a apresentação do método de subgradiente utilizado na resolução do problema dual Lagrangeano.

5.2. Problema Relaxado e Dual Lagrangeano

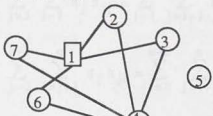
Retome-se a formulação apresentada para o *m-TSP* no capítulo 2 (pg. 8). No método que se apresenta ([21]) considera-se a relaxação das **n-1** restrições (4), que impõem que o grau de cada vértice, com excepção do inicial, seja igual a dois. Resulta assim, um problema - problema relaxado - possível de resolver em tempo de execução polinomial, conforme pretendido.

Considerando esta relaxação, adiciona-se à formalização apresentada **(P)** a restrição:

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} x_{ij} = n - 1 \qquad (8)$$

que, sendo redundante na formalização **(P)**, deixa de o ser quando relaxadas as restrições (4). Com a inclusão de (8), qualquer solução admissível do problema relaxado contém **n-1+m** arestas (**m** resultantes das restrições (3) e **n-1** desta) igual ao número de arestas de qualquer solução admissível do *m-TSP*. Assim, na relaxação apenas se viola o facto de cada cidade, com excepção da inicial, ser visitada exactamente uma vez por um só caixeiro (ou seja, o grau de cada vértice ser dois), podendo ter-se uma solução como a do seguinte exemplo:

Ex: **n = 7** e **m = 2**,



Solução admissível para o problema relaxado e não admissível para o *m-TSP*

Fig. 5.1.

Considere-se então a relaxação das restrições (4), e a inclusão de (8). Multiplicando as **n-1** restrições (4) por um vector de multiplicadores de Lagrange :

$$\pi = [\ \pi_2\ ,\ldots,\ \pi_n\ ]$$

e adicionando o resultado à função objectivo, pode penalizar-se a sua violação.

Assim, tem-se para cada $j \in S$ ($j = 2,\ldots, n$) :

$$\pi_j \left[2 - \left(x_{j1} + \sum_{i<j} x_{ij} + \sum_{i>j} x_{ji} \right) \right]$$

j-ésima restrição relaxada

O problema relaxado é então dado por:

(LRP)

$$L(\pi) = \text{Min} \left\{ \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} x_{ij} + \sum_{i=2}^{n} c_{i1} x_{i1} + \sum_{j=2}^{n} \pi_j \left[2 - \left(x_{j1} + \sum_{i<j} x_{ij} + \sum_{i>j} x_{ji} \right) \right] \right\} \qquad (9)$$

s.a: (2), (3), (5 - 8)

onde π_j , $j = 2,\ldots, n$ são livres dado serem multiplicadores associados a restrições de igualdade.

Reordenando os termos de $L(\pi)$ tem-se:

$$L(\pi) = \text{Min} \left\{ \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} x_{ij} + \sum_{i=2}^{n} c_{i1} x_{i1} + 2 \sum_{j=2}^{n} \pi_j - \sum_{j=2}^{n} x_{j1} \pi_j - \sum_{j=2}^{n} \pi_j \sum_{i=1}^{j-1} x_{ij} - \sum_{j=2}^{n-1} \pi_j \sum_{i=j+1}^{n} x_{ji} \right\}$$

como :

$$\sum_{j=2}^{n} \sum_{i=1}^{j-1} \pi_j x_{ij} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \pi_j x_{ij}$$

$$\sum_{j=2}^{n-1} \sum_{i=j+1}^{n} \pi_j x_{ji} = \sum_{i=2}^{n-1} \sum_{j=i+1}^{n} \pi_i x_{ij}$$

fazendo : $\tilde{c}_{ij} = c_{ij} - \pi_i - \pi_j$ com $\pi_1 = 0$ (*)

(*) Nota: Prova-se fácilmente que se C é simétrica $\tilde{C}$ também é. Logo, o problema a resolver permanece com a matriz de distâncias simétrica.

vem:

(LRP)

$$L(\pi) = \min_{x} \left\{ \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \tilde{c}_{ij} x_{ij} + \sum_{i=2}^{n} \tilde{c}_{i1} x_{i1} + 2 \sum_{j=2}^{n} \pi_j \right\} \quad (10)$$

s.a:

$$\sum_{j \in S} x_{1j} = m \quad (2)$$

$$\sum_{i \in S} x_{i1} = m \quad (3)$$

$$\sum_{\substack{i,j \in S_k \\ i<j}} x_{ij} \leq |S_k| - 1 \quad \forall S_k \subseteq S, \; S_k \neq \emptyset \quad (5)$$

$$x_{ij} = 0,1 \quad 1 \leq i < j \leq n \quad (6)$$

$$x_{i1} = 0,1 \quad \forall i \in S \quad (7)$$

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} x_{ij} = n-1 \quad (8)$$

A solução do problema relaxado fornece, para um dado vector de multiplicadores de Lagrange, um minorante para o valor óptimo do *m-TSP*, ou seja:

Teor. ([22]): $L(\pi) \leq z^*$ onde π é um certo vector de multiplicadores de Lagrange e z^* o valor da solução óptima do problema (**P**).

O valor deste minorante, estando relacionado com os valores dos multiplicadores, pode ser melhorado optimizando os valores de π, o que se consegue com a resolução do problema Dual Lagrangeano definido por:

$$\textbf{(DL)} \qquad L(\pi^*) = \max_{\pi} \{L(\pi)\} \quad (11)$$

A resolução de **(DL)** não oferece em geral a solução do *m-TSP* **(P)**, sendo contudo válida a seguinte relação como caso particular do teorema anterior:

Cor.: $L(\pi^*) \leq z^*$.

As restrições do problema **(LRP)** - (2), (3), (5-8) - originando um conjunto discreto de pontos, dificultam a resolução de **DL** devido a $L(\pi)$ (função objectivo de **(LRP)**) não ser diferenciável em toda a parte. Sabendo-se que a resolução do problema relaxado, para um certo vector de multiplicadores, fornece um subgradiente da função objectivo $L(\pi)$, tal facto pode ser contornado pelo recurso a métodos de optimização por subgradiente para a resolução do dual Lagrangeano ([22]). O método utilizado é descrito no ponto 5.4., sendo préviamente explicado o algoritmo para a determinação do valor do problema relaxado, para um certo vector de multiplicadores de Lagrange.

5.3. Resolução do Problema Relaxado

Considere-se o problema relaxado **(LRP)** atrás definido (pg. 39). Na formulação pode observar-se que as variáveis x_{1j}, $i = 2,...,n$ se encontram isoladas num dos membros da função objectivo, bem como nas restrições (3) e (7). O mesmo acontece no que diz respeito às variáveis x_{ij} , $i = 1,..., n-1$; $j = 2,..., n$ relativamente às restrições (2), (5), (6), e (8). Resultam então dois subproblemas de fácil resolução, podendo o valor do minorante ser calculado com base no valor do solução fornecida por cada um dos subproblemas.

É fácil notar que a parte respeitante às variáveis x_{ij} de **(LRP)**, ou seja:

$$\text{Min} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \tilde{c}_{ij} x_{ij}$$

s.a. : (2), (5), (6) e (8)

traduz o problema da determinação de uma árvore de suporte com restrições de grau no vértice inicial, **1**, pois a restrição (2) impõe que **grau (1) = m**, de menor custo possível (*m-SST*).

Por outro lado, e no que concerne às variáveis x_{i1} tem-se:

$$\text{Min} \sum_{i=2}^{n} \tilde{c}_{i1}\, x_{i1}$$

s.a. : (3) e (7)

representando a determinação das **m** arestas de menor custo incidentes no vértice **1**, as quais dir-se-á, representarem as arestas de retorno dos caixeiros viajantes à cidade inicial.

Deste modo, o valor do problema relaxado, para um dado vector de multiplicadores de Lagrange, pode ser encontrado com o algoritmo que se descreve de seguida:

Algoritmo (III) : Determinação do valor de **(LRP)**;

Dados: Matriz de custos transformados, $\tilde{C}$;
Vector de multiplicadores de Lagrange, π;

Resultado: Valor do problema relaxado para π, $L(\pi)$;

1. Determinar, em **G = (V,A)** utilizando a matriz de custos transformados $\tilde{C}$, uma árvore de suporte onde o vértice **1** tenha grau **m**, de menor custo possível (utilizou-se o Algoritmo (I) - cap. 3 - com **b = m** e $\tilde{C}$);
   Seja **T** a *m-SST* e **csst** (π) o seu custo;

2. Seleccionar **m** arestas que representem o retorno dos caixeiros, utilizando $\tilde{C}$.
   Seja **cret** (π) a soma dos custos destas arestas;

3. Calcular o valor do problema relaxado, fazendo :

$$L(\pi) = \text{csst}\ (\pi) + \text{cret}\ (\pi) + 2 \sum_{j=2}^{n} \pi_j .$$

O passo **2** deste algoritmo consiste na identificação das **m** arestas de menor custo incidentes em **1**. No entanto, esta escolha pode originar um conjunto de arestas que conjuntamente com as da *m-SST*, **T**, formem **m** subcircuitos imediatos, ou seja, circuitos da forma $\{(1,j), (j,1)\}$. Sabendo-se que qualquer solução admissível para o *m-TSP* tem no máximo **m-1** subcircuitos imediatos, desde que **n > m-1**, o valor do minorante para o *m-TSP*, obtido com a resolução do problema relaxado, pode ser melhorado sempre que na junção de arestas referida haja **m** subcircuitos imediatos. Neste caso as **m** arestas representativas do retorno dos caixeiros serão as **m-1** de menor custo incidentes em **1** e a **m+1**-ésima nas mesmas condições, em vez das **m** de menor custo.

Procedendo desta forma, consegue-se encontrar, sem grande esforço computacional, um minorante para o valor do *m-TSP* nunca inferior ao que se obteria com a escolha das **m** arestas de menor custo incidentes em **1** ([21]).

Logo, a escolha das arestas de retorno é feita da seguinte forma:

2A. Seleccionar as **m** arestas de menor custo incidentes em **1**;

2B. Se as **m** arestas seleccionadas conjuntamente com as da árvore **T**, encontrada em **1**, formarem no máximo **m-1** subcircuitos imediatos ir para **3**;
C.c., substituir a **m-ésima** aresta de menor custo, ou seja a de maior custo entre as seleccionadas em **2A**, pela **m+1-ésima** de menor custo incidente em **1**.

5.4. Método de Subgradiente

Na presente secção descreve-se o método de optimização subgradiente utilizado na resolução do problema Dual Lagrangeano, anteriormente definido como sendo:

$$\text{(DL)} \qquad L(\pi^*) = \max_{\pi} \{L(\pi)\} \qquad (11)$$

Para a actualização dos valores dos multiplicadores é necessário o cálculo de um majorante para o *m-TSP*, tendo neste trabalho sido utilizado o fornecido pela heurística de *Frieze* (Algoritmo (II) - cap. 4).

O primeiro passo para a determinação dos multiplicadores de Lagrange consiste no cálculo das direcções de subgradiente que, estando relacionadas com a violação de cada uma das restrições relaxadas, são na k-ésima iteração dadas por:

$$\gamma_j^k = 2 - \left(x_{j1} + \sum_{i=1}^{j-1} x_{ij} + \sum_{i=j+1}^{n} x_{ji} \right) \qquad \forall\, j \in S \qquad (12)$$

indicando assim, de quanto foi violada cada uma das restrições relaxadas na solução obtida para o problema relaxado. O vector subgradiente, γ, embora nem sempre originando um melhor valor para o minorante, indica sempre o semi-espaço que contém o óptimo.

A actualização dos valores dos multiplicadores de Lagrange foi feita à custa dos valores anteriores e das direcções de subgradiente, de acordo com a fórmula:

$$\pi_j^{k+1} = \pi_j^k + t_k \gamma_j^k \qquad (13)$$

onde: $t_k = \lambda_k \left(\bar{z} - L(\pi^k) \right) / \| \gamma^k \|^2 \qquad (14)$

em que: $L(\pi^k)$ é o valor obtido para o problema relaxado na k-ésima iteração;
$\| . \|$ é a norma Euclidiana do vector γ;
$\bar{z}$ é o valor do majorante fornecido pela heurística de Frieze;
λ_k é um escalar;

Um dos maiores problemas na utilização dos métodos de subgradiente está na escolha dos valores de λ_k determinantes do passo do algoritmo t_k. Os valores de $\{\lambda_k\}$ foram fixados de acordo com a seguinte regra empírica: inícia-se com o valor $\lambda_0 = 1$, o qual se divide ao meio sempre que em cinco iterações sucessivas não se obtenha melhoria no valor da solução do problema relaxado.

Esta regra, semelhante à proposta por *Held et al.* [28], é a que tem proporcionado melhores resultados práticos, embora não seja possível demonstrar a sua convergência ([22]).

O valor inicial de λ (=**1**) foi escolhido após a realização de diversos testes em que se compararam os resultados obtidos com λ_0 = **.25** ; **.5** ; **1** ; **2**. Nestes observou-se que, na maior parte dos casos em que o melhor valor para λ_0 era **1** a escolha de outro qualquer dos valores originava valores significativamente piores para os minorantes. Por outro lado, quando a melhor inicialização não era **1**, os minorantes obtidos com $\lambda_0 = 1$ não se encontravam muito afastados dos obtidos com a melhor das inicializações.

Foram igualmente realizados testes para encontrar o número de iterações sem melhoria para o minorante e em que não se deveria dividir λ. Nos exemplos experimentados λ foi dividido ao fim de **k = 1, 5, 10, 15, 20** iterações sucessivas sem melhoria no valor do minorante, tendo sido os melhores resultados obtidos para **k = 5**.

Nos testes computacionais elaborados, cujos resultados se apresentam no capítulo 7, foram impostas condições de paragem relacionadas com os valores dos minorantes obtidos, com o número de iterações e com o valor de λ_k como se verá no algoritmo que se apresenta.

A resolução do problema dual Lagrangeano foi então feita com base no método de subgradiente descrito no algoritmo seguinte:

Algoritmo (IV) :

Dados: Matriz de custos, **C**;
Valor mínimo pretendido para o majorante do erro relativo, ε;
Uma solução admissível e o respectivo valor, representando um majorante para o valor óptimo do problema, $\overline{z}$ (Algoritmo (II) - cap.4);

Resultado: Minorante para o valor óptimo do problema, $\underline{z}$;

1. Inicializar o vector de multiplicadores de Lagrange, fazendo ([21]):

$$\pi_1^0 = 0$$

$$j \in S \quad \pi_j^0 = \text{Min} \{ c_{ij} , \ i = 1,...,n \ ; \ i \neq \ell \}$$

onde ℓ é o vértice a menor distância de **j**, i.é.:

$$c_{\ell j} = \text{Min} \{ c_{ij} \ ; i = 1,..., n \};$$

Fazer : $\lambda \leftarrow 1$;

$k \leftarrow 0$; {contador do nº de iterações}

$k1 \leftarrow 0$; {contador do nº de iterações em que o valor do minorante não é alterado e λ não é dividido}

$k2 \leftarrow 0$; {contador do nº de iterações em que o valor do minorante não é alterado}

$max\_iter \leftarrow 100$; { nº máximo de iterações permitidas}

$\underline{z} \leftarrow -\infty$; {valor do minorante}

2. Determinar, para o vector de multiplicadores de Lagrange π^k, o valor do problema relaxado (Algoritmo (III) - 5.3. - com $\tilde{c}_{ij} = c_{ij} - \pi_i^k - \pi_j^k$). Seja $L(\pi^k)$;
Se $L(\pi^k) > \underline{z}$ fazer $\underline{z} \leftarrow L(\pi^k)$;

3. Terminar se:

(i) $L(\pi^k) = \overline{z}$ se se está na raíz da árvore a solução admissível encontrada pelo algoritmo (II) é óptima [a] ;

(ii) $(\overline{z} - \underline{z}) / \overline{z} \leq \varepsilon$ o valor do majorante é suficientemente próximo do do minorante;

C.c. ir para **4**;

4. Se, $L(\pi^k) \leq \underline{z}$ Fazer: $k1 \leftarrow k1 + 1$; $k2 \leftarrow k2 + 1$;
C.c. Fazer: $k1 \leftarrow 0$; $k2 \leftarrow 0$;

$k \leftarrow k + 1$;

Parar se, $k > max\_iter$;

5. Se $k1 = 5$ Fazer: $\lambda \leftarrow \lambda/2$; $k1 \leftarrow 0$;

Calcular as direcções de subgradiente (fórmula 12) e o quadrado da norma do vector γ;

Parar se:

$\| \gamma \| = 0$ o valor do minorante coincide com o valor óptimo do problema (b);

$k2 = 15$ foram realizadas quinze iterações sem melhorar o valor do minorante;

$\lambda < .001$ o minorante não é melhorado para valores pequenos de λ;

C.c. continuar;

6. Calcular t_k (fórmula 14);
Actualizar o vector de multiplicadores de Lagrange (fórmula 13);
Voltar a **2.**

Observação: Como se verá no capítulo 6, este procedimento vai ser integrado num "branch and bound" (b&b). No b&b ele será usado quer para o problema inicial quer para subproblemas resultantes de ramificações. Neste último caso terão de ter-se em consideração alguns aspectos, nomeadamente, o significado dos critérios de paragem ((a) e (b) nos passos **3** e **5**, respectivamente) e a inicialização dos valores dos multiplicadores de Lagrange (passo **1**).

A inicialização dos valores dos multiplicadores de Lagrange (em **1**), proposta por *Gavish & Srikanth*, foi comparada com a inicialização a zero e ao mínimo de cada linha da matriz de custos, sendo a utilizada a que proporcionou melhores resultados.

6. Método de Partições e Avaliações Sucessivas

6.1. Introdução

A solução óptima do problema foi determinada com recurso a um método de partições e avaliações sucessivas (b&b - " branch and bound"). As primeiras referências a este tipo de métodos surgiram na década de 50, sendo de salientar os trabalhos de *Dantzig et al.* [11, 12] e o de *Eastman* [15] para o *TSP* simétrico. O termo "branch and bound" surge uns anos mais tarde com um algoritmo para o *TSP* apresentado por *Little et al.* [37].

Os métodos de pesquisa em árvore consistem na partição do problema inicial, $\mathbf{P_0}$, em subproblemas, $\mathbf{P_1}$, $\mathbf{P_2}$,..., $\mathbf{P_k}$ (que em conjunto representam $\mathbf{P_0}$), que possam ser mais fácilmente resolvidos. Entende-se por resolução dum subproblema:

- determinar a sua solução óptima;

ou

- mostrar que o valor da sua solução óptima é pior que o da melhor solução admissível, encontrada até ao momento;

ou

- concluir que é um problema impossível.

A partição é representada por uma árvore, em que cada nodo está associado a um subproblema (Fig. 6.1). A raíz da árvore diz respeito ao problema inicial, $\mathbf{P_0}$.

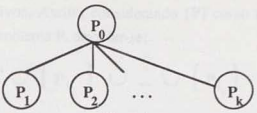


Fig. 6.1

A partição do problema inicial é justificada pelo facto dos subproblemas resultantes serem de mais "fácil" resolução, pois de dimensões menores, podem ainda ter estruturas mais simples, não partilhadas por P_0.

Porém, uma só partição, pode ainda originar subproblemas de "difícil" resolução. Assim, para cada um destes subproblemas, P_i, é feita uma nova partição, num conjunto de subproblemas menores, $P_{i_1}, P_{i_2}, ..., P_{i_r}$ (Fig. 6.2).

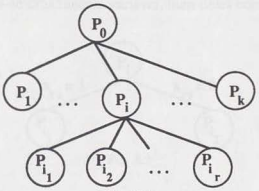


Fig. 6.2

Estas partições são repetidas para cada um dos subproblemas que não possa ser resolvido.

Os nodos da árvore de grau um ou representam subproblemas que ainda não foram resolvidos - designando-se por nodos pendentes - ou contém a informação da solução desse subproblema.

Para que a solução óptima de P_0 seja garantidamente encontrada por estes métodos, é necessário que os subproblemas criados englobem todas as hipóteses de soluções para P_0. Por outro lado deve ser impedida a consideração de subproblemas idênticos, para garantir que a solução óptima de P_0 é solução dum e dum só subproblema, caso não

existam óptimos alternativos. Assim, considerando {P} como o conjunto de todas as soluções admissíveis do problema P, deve ter-se:

$$\{P_i\} = \{P_{i_1}\} \cup \{P_{i_2}\} \cup \ldots \cup \{P_{i_r}\}$$

$$\{P_{i_s}\} \cap \{P_{i_q}\} = \emptyset, \quad s \neq q$$

A partição, pode ser binária ou múltipla. No primeiro caso cada subproblema é subdividido em apenas dois subproblemas, enquanto no segundo é subdividido em dois ou mais.

No método utilizado neste trabalho, proposto por *Gavish & Srikanth* [21], optou-se pela partição binária. Assim, nos dois novos subproblemas que se consideram, a partir de um nodo da árvore, fixa-se uma mesma variável, num deles com o valor um e no outro com o valor zero (Fig 6.3).

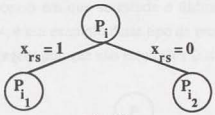


Fig. 6.3

À medida que se "desce" na árvore, vão-se tendo subproblemas em que o número de variáveis fixadas (em um ou zero) é progressivamente maior, ou seja, vão-se obtendo problemas de dimensão progressivamente menor. Tal leva a que, a certa altura, dado o número de variáveis já fixadas, seja possível a determinação da solução do subproblema resultante.

A regra de selecção, ou seja, a escolha da variável a fixar, bem como o ramo a seguir, não é única. Neste capítulo é apresentada a que se utilizou, sendo ainda referidas algumas alternativas.

Para além do tipo de partição e da regra de selecção utilizada, os métodos de pesquisa em árvore, diferenciam-se ainda pelo tipo de procura. A procura pode ou não incorporar

informação subjacente aos subproblemas em nodos pendentes. Quando não incorpora, consideram-se os dois tipos seguintes:

(1) Em profundidade;

(2) Em largura;

Estes, quando considerados isoladamente designam-se por estratégias de procura puras. Na resolução de problemas de optimização combinatória, não é usual o recurso à procura em largura pura, sendo mais frequente o uso de estratégias mistas com incorporação de informação.

(1) Procura em profundidade

Na procura em profundidade, a partição é feita a partir do último subproblema gerado, até ser atingido um que possa ser resolvido. Após a resolução deste subproblema, começa-se o passo de retrocesso em que se estuda o último subproblema num nodo pendente da árvore. A Fig. 6.4, é um exemplo deste tipo de procura, em que a numeração dos subproblemas indica a ordem por que são estudados (está a supôr-se que se utiliza partição binária).

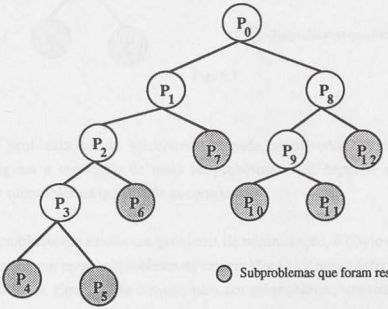


Fig. 6.4

Neste exemplo foi dada prioridade ao sucessor esquerdo sobre o sucessor direito. Tal não é porém necessário.

(2) Procura em largura

O estudo é feito de nível para nível, em que o nível 0 da árvore representa o problema P_0 e o nível k contém os subproblemas gerados a partir dos subproblemas no nível k-1. P_0 é subdividido nos subproblemas P_1, P_2 (ou $P_1, P_2, ..., P_j$ se não se utilizar pesquisa binária) no nível 1 da árvore e cada um destes é estudado antes de se passar ao nível 2. Os subproblemas do nível 1 que não possam ser resolvidos são subdivididos em subproblemas no nível 2, que são investigados antes de se passar ao nível 3. Repete-se o processo até não existirem nodos pendentes. A Fig. 6.5, exemplifica este tipo de procura, em que os números dos subproblemas indicam a ordem por que foram analisados.

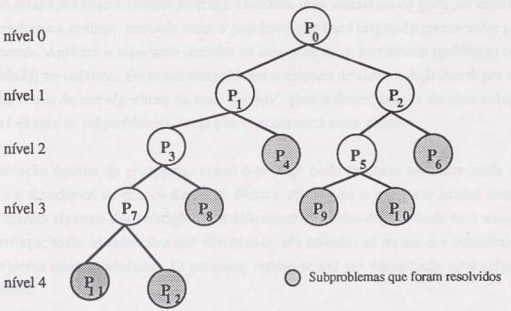


Fig. 6.5

Assim, o problema que se selecciona depende do método de procura utilizado, podendo originar a resolução de mais subproblemas. Tal depende dos valores das soluções dos subproblemas que se vão encontrando.

Sendo o problema em estudo um problema de minimização, é óbvio que, as soluções admissíveis a que um certo subproblema dá origem têm valor nunca inferior ao minorante desse subproblema. Então, se se obtiver, para um subproblema, um minorante de valor superior ao da melhor solução admissível, encontrada até ao momento, pode ser cancelada a ramificação a partir desse nodo. Pois as soluções admissíveis que se iriam encontrar, não podem representar a solução óptima, dado já se dispôr de uma melhor. Tal justifica a importância da determinação de minorantes para cada subproblema.

Utilizando métodos diferentes de partição, de selecção, de procura e de determinação de minorantes e de majorantes outros autores, que se passam a referir, resolvem o *m-TSP* por métodos do tipo b&b.

Jonker & Volgenant [32] resolvem por um algoritmo de tipo b&b, o *TSP* resultante da transformação mensionada, utilizando no cálculo de minorantes a relaxação referida no capítulo anterior. Com base nos resultados computacionais que apresentam, comparam o seu método com o de *Gavish & Srikanth* [21].

Ali & Kennington [1] apresentam, um método de b&b, para o caso assimétrico, com base na relaxação anteriormente referida. Utilizam uma estratégia de partição múltipla escolhendo para analisar, em cada caso, o problema que tiver originado menor valor para o minorante. Aplicam o algoritmo também ao caso simétrico, resolvendo problemas com cem cidades no máximo. Estes autores criticam o método de *Gavish & Srikanth* por não permitir o uso de um algoritmo de tipo "greedy" para a determinação de uma solução admissível para os subproblemas, facto que se comentará neste ponto.

A solução óptima de problemas como o *m-TSP* pode também ser encontrada por recurso a algoritmos de planos de corte. Nestes, resolve-se o problema inicial sem se considerarem algumas das restrições que dificultam a sua resolução. Dada uma solução que verifique todas as restrições não eliminadas, são testadas as restantes e introduzidas no problema quando violadas. O processo repete-se até ser encontrada uma solução admissível.

Laporte & Nobert [34], estudando o caso em que **m** é variável, recorrem a um método de planos de corte para a resolução do *m-TSP*, apresentando e comparando dois algoritmos. Em ambos, são eliminadas as restrições que impedem a formação de subcircuitos "ilegais" (ou seja subcircuitos que não possam fazer parte de uma solução admissível do *m-TSP*) e as de integralidade. Diferenciam-se por, num deles (o qual designam por "*straight algorithm*") só serem consideradas as restrições de eliminação de subcircuitos após ter sido encontrada uma solução inteira, enquanto que no outro ("*reverse algorithm*") é procurada em primeiro lugar uma solução que não contenha subcircuitos "ilegais" e só depois consideradas as restrições de integralidade.

Orloff [40] transforma o *m-TSP* num *TSP* e propõe o algoritmo de planos de corte de *Bellmore & Malone* [3] em que se eliminam apenas as restrições que impedem a formação de subcircuitos, provando que este algoritmo resolve o problema em estudo.

Neste capítulo começa-se por descrever a forma como é escolhida a variável a fixar em cada nodo da árvore - regra de selecção. Com o objectivo de tentar fixar implicitamente variáveis é feita uma análise sensitiva às soluções dos problemas relaxados em cada nodo da árvore, que se explica no ponto seguinte. Por fim, é descrito o método utilizado e apresentado o algoritmo global (*).

6.2. Regra de Selecção

A escolha da variável a fixar em cada nodo da árvore, bem como o seu valor, afecta a eficiência de algoritmos do tipo b&b. Torna-se, então conveniente escolher as variáveis mais credíveis de pertencerem/não pertencerem a uma solução admissível do *m-TSP*, fixando-as com valor um, caso seja "elevada" a credibilidade de pertencerem e a zero no caso contrário.

A escolha foi feita de acordo com a regra ([21]) que se justifica neste ponto, sendo préviamente apresentadas algumas notações.

Sejam:

$A_{(LRP)} = \{(i_1, j_1), (i_2, j_2), ..., (i_{n+m-1}, j_{n+m-1})\}$ o conjunto das arestas numa solução do problema relaxado;

$x_{i_k j_k}$ a variável correspondente à aresta $(i_k, j_k) \in A_{(LRP)}$;

p_k alterações no valor da solução se $x_{i_k j_k}$ é fixada com valor zero;

q_k alterações no valor da solução se $x_{i_k j_k}$ é fixada com valor um;

N_k número de arestas de $A_{(LRP)}$ que terão de ser retiradas se $x_{i_k j_k}$ fôr fixada com valor um.

(*) Nota: No que se segue, com o objectivo de simplificar o texto, fala-se de fixação de arestas e de variáveis sempre com o mesmo significado, ou seja representando sempre fixação de variáveis.

Os valores de p_k podem ser determinados de acordo com o método que se apresenta no ponto seguinte (§ 6.3.1.; § 6.3.2.).

Para o cálculo de q_k, note-se que fazer $x_{i_k j_k} = 1$ só altera a solução se, estando livre, existirem arestas incidentes em i_k e/ou j_k anteriormente fixadas, com valor um. Os valores de q_k podem determinar-se a partir dos de N_k, $1 \le k \le n+m-1$, sendo estes fácilmente encontrados para uma dada solução do problema relaxado. Assim, tem-se:

i) $N_k = 0 \Rightarrow q_k = 0$, pois a fixação a um da **k**-ésima variável não origina alterações na solução;

ii) $N_k = 1 \Rightarrow q_k = p_\ell$; onde $x_{i_\ell j_\ell}$ é a variável que tem de ser nula se $x_{i_k j_k} = 1$;

iii) $N_k > 1$ tem que ser encontrada uma nova solução para o problema relaxado, pois sendo fixada a um a **k**-ésima variável, pelo menos duas arestas têm que ser retiradas da actual solução.

Como é óbvio, se $i_k = 1$, N_k só depende do número de arestas incidentes em j_k.

Ex.: Suponha-se que num problema com **8** cidades e **2** caixeiros, se obteve, num nodo da árvore, a solução do problema relaxado:

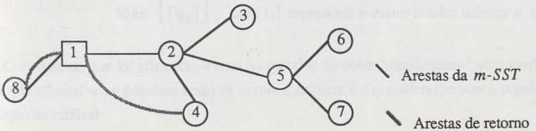


Fig. 6.6

em que estão fixadas a um as arestas **(1,2)** e **(5,7)**

i) $q_{(1,4)} = 0$ ($N_{(1,4)} = 0$), pois se se fixar a um a aresta **(1,4)**, a solução não é alterada, dado não existirem arestas incidentes em **4** fixadas a um;

ii) $N_{(5,6)} = 1$, se **(5,6)** fôr fixada a um, como $x_{5,7} = 1$ e não pode haver mais de duas arestas incidentes em cada vértice de **S**, a aresta **(2,5)** tem que

ser fixada a zero. Assim, a alteração no valor da solução por fixar a um **(5,6)**, coincide com a alteração no mesmo valor por fixar a zero **(2,5)**. Logo, $q_{(5,6)} = p_{(2,5)}$;

iii) $N_{(2,3)} = 2$, pois como **(1,2)** está fixada a um, as arestas **(2,5)** e **(2,4)** não podem pertencer à solução em que se fixa a um **(2,3)**. Logo, para calcular $q_{(2,3)}$ é necessário encontrar uma nova solução para o problema relaxado, em que não se permita mais nenhuma aresta incidente em **2**, para além de **(2,3)** e **(1,2)**;

Para calcular $q_{(2,5)}$ ($N_{(2,5)} = 3$), dado **(1,2)** e **(5,7)** estarem fixadas a um, as arestas **(2,3)**, **(2,4)** e **(5,6)** terão de ser fixadas a zero se **(2,5)** fôr fixada a um. Assim, é necessário encontrar uma nova solução que inclua apenas as arestas **(1,2)**, **(2,5)** incidentes em **2** e **(2,5)**, **(5,7)** incidentes em **5**.

Os valores de p_k e q_k são calculados para se poderem identificar as variáveis com maior credibilidade de serem incluídas/excluídas da solução do *m-TSP*. Se q_k é "elevado" faz sentido pensar que $x_{i_k j_k}$ não deve pertencer à solução. Por outro lado, se p_k é "alto" é credível pensar que $x_{i_k j_k}$ deve pertencer à solução.

Assim, selecciona-se para fixar ([21]), em cada nodo da árvore a variável $x_{i_k j_k}$ que corresponder a:

$$\text{Máx}\left\{\lceil q_k \rceil\right\}, \quad \text{onde } \lceil . \rceil \text{ representa o maior inteiro inferior a } q_k.$$

Com base no que foi afirmado, e com o objectivo de obter "rápidamente" uma melhor solução admissível, o próximo nodo da árvore a analisar é o que corresponder à seguinte fixação da variável:

$$x_{i_k,j_k} = \begin{cases} 1 & \text{se } \lceil q_k \rceil \leq \lceil p_k \rceil \\ 0 & \text{c.c.} \end{cases}$$

Como alternativa a este método, poderia fixar-se a variável que verificasse $\text{Máx}\left\{\lceil p_k \rceil\right\}$, sendo, em primeiro lugar, analisado o problema que correspondesse à expressão anterior.

Uma outra hipótese, seria fixar a variável correspondente à aresta incidente nos vértices cujos graus sejam mais violados, ou seja, cuja soma dos graus é máxima. Assim, fixar-se-ia, a um, a variável $x_{i_k j_k}$:

$$\text{Máx} \left\{ [\ 0;\ \text{grau}(i_k) - 2] + [0;\ \text{grau}(j_k) - 2] \right\} \quad \text{se} \quad i_k \neq 1\ ;\ j_k \neq 1;$$

$$\text{Máx} \left\{ [\ 0;\ \text{grau}(i_k) - 2] \right\} \quad \text{se} \quad j_k = 1;$$

$$\text{Máx} \left\{ [0;\ \text{grau}(j_k) - 2] \right\} \quad \text{se} \quad i_k = 1;$$

Esta última regra é a que envolve menos tempo de execução, se $\exists_k : N_k > 1$. Embora a escolha da variável a fixar seja, neste caso, mais "rápida", pode ter como consequência que sejam necessários mais nodos na árvore o que torna o tempo total pior. Seria, no entanto, interessante a comparação das diversas regras.

6.3. Fixação Implícita de Variáveis

No ponto anterior justificou-se o critério utilizado para fixação explícita de variáveis. Nos processos de ramificação em árvore é vantajoso analisar a solução do problema relaxado de forma a tentar, em cada nodo da árvore, fixar implicitamente variáveis conseguindo-se assim, diminuir o número de níveis da árvore a partir do nodo em análise. Esta fixação foi feita de acordo com dois critérios, um baseado nas restrições de grau e outro com base numa análise dos custos.

Relativamente ao primeiro destes critérios são analisados os graus dos vértices que correspondem a cada variável que se fixa. Assim, sempre que uma variável x_{ij} é fixada com valor um vê-se se existe alguma outra variável incidente em **i** (**j**) anteriormente fixada a um e, em caso afirmativo fixam-se todas as restantes variáveis incidentes em **i** (**j**) a zero, pois qualquer solução admissível para o *m-TSP* tem duas e só duas arestas incidentes em cada vértice, com excepção do inicial. Se **i=1**, faz-se um estudo análogo, fixando a zero as restantes variáveis incidentes em **1**, caso já existam fixadas com valor um **2m** variáveis. Se x_{ij} é fixada com valor zero, contam-se o número de variáveis fixadas a zero

incidentes em **i** (**j**) e caso seja **n-3** fixam-se as duas restantes a um. Tal como no primeiro caso, se **i=1**, só se fixam as restantes a um quando o número de variáveis incidentes em **1**, fixadas a zero fôr **n-m-2**, pois qualquer solução admissível para o *m-TSP* tem no mínimo **m+1** arestas incidentes em **1**, caso em que **m-1**têm que ser duplicadas.

O segundo critério engloba um estudo da alteração no valor da solução do problema relaxado se uma variável, ainda livre, fôr retirada/incluída da referida solução, que se designou por análise sensitiva. Uma variável, ainda livre, pode ser fixada a um se fazendo parte da solução do problema obtida, o acréscimo no valor da solução - que se designará por penalidade - originado por a retirar da solução fôr "elevado". Ou seja, se o valor do problema relaxado obtido com a imposição $x_{ij} = 0$ fôr superior ao do majorante, tem-se a garantia de que x_{ij} tem que ter valor **1** na melhor solução admissível do subproblema em estudo. Neste caso, x_{ij} é fixada com valor um. Relativamente a uma variável livre e não pertencente à solução do problema relaxado, pode ser feito um estudo análogo na tentativa de a fixar a zero. Assim, uma variável nestas condições, pode ser fixada com valor zero se fôr elevada a penalidade originada pela sua inclusão na solução. Dada uma solução do problema relaxado é, então, feito este estudo para cada uma das variáveis livres, tentando fixar a um as que pertençam à solução e a zero as restantes.

Com o objectivo de detalhar o estudo desta análise sensitiva torna-se conveniente relembrar que o método utilizado no cálculo de minorantes fornece, em cada nodo da árvore, uma solução com base numa árvore de suporte, onde o vértice **1** tenha grau **m**, de custo mínimo e **m** arestas representativas do retorno dos caixeiros à cidade de partida.

No parágrafo seguinte é introduzido e justificado um algoritmo para a análise sensitiva às variáveis da *m-SST*. Seguir-se-à o estudo efectuado às **m** variáveis de retorno, sendo por fim apresentada a análise efectuada às variáveis que não pertençam à solução.

6.3.1. Análise Sensitiva às Variáveis da *m-SST*

Para uma variável x_{ij} que pertença à *m-SST*, e tendo como objectivo determinar uma solução do problema relaxado que não a inclua, poderia encontrar-se nova *m-SST* onde $c_{ij}=\infty$. Porém, sendo **n-1** estas variáveis e $O(|A| \log|V| + |V| \log|V|)$ a complexidade do algoritmo implementado, para este estudo resultaria uma complexidade da ordem de n^3.

Gavish & Srikanth [20], provando que as arestas que formam as novas árvores pertencem todas a uma mesma *m-SST*, apresentam e demonstram a validade de um algoritmo, da ordem de n^2, para a determinação simultânea de todas as arestas que substituem cada uma das que se pretendem excluir da *m-SST*.

Começam por apresentar um método para a análise sensitiva às variáveis de uma *SST*, que será necessário ao estudo que se pretende efectuar, sendo portanto justificado em primeiro lugar. Segundo se pensa, este caso foi o único anteriormente abordado, por *Spira & Pan* [43] e por *Chin & Houck* [8], não tendo sido generalizado a árvores com restrições de grau.

(a) Análise às arestas de uma *SST*

Seja $T = (V, A_T)$ e $A_T = \{a_1, a_2, ..., a_{n-1}\}$. Se T fôr uma *SST* em $G = (V, A)$, uma *SST* em $G_i' = (V, A\backslash\{a_i\})$ difere de T numa só aresta pertencente a uma *SST*, **T"**,em $G" = (V, A_T")$, onde $A_T" = A\backslash A_T$.

Este resultado é facilmente compreensível, se se pensar que as arestas de uma *SST* num grafo são as de menor peso incidentes em cada vértice do grafo, que não originam ciclos. Deste modo, retirada a aresta a_i de A e encontrada uma *SST* em G_i', ela inclui forçosamente todas as arestas de $A_T\backslash\{a_i\}$, pois representam as arestas de menor custo incidentes nos vértices de V que não originam ciclo, e a aresta de menor peso que liga as duas subárvores resultantes de $A_T \backslash \{a_i\}$. Esta nova aresta pertence óbviamente a **T"**. É importante salientar que nem todas as arestas de **T"** substituem arestas de **T**, ou seja,

pertencem a uma árvore de um dos G_1'. Para melhor compreensão considere-se o seguinte exemplo:

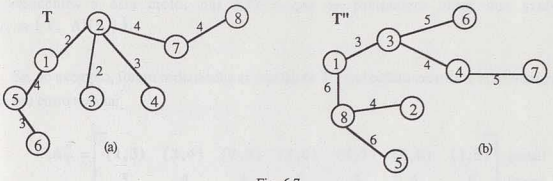


Fig. 6.7

Como **T** é uma *SST* em **G** e **T"** uma *SST* em **G"**, tem-se a garantia que todas as restantes arestas incidentes em cada vértice de **V** ou têm custo superior ou igual às de **T"**, ou originam um ciclo em **T"**.

Se se retirar de **T** a aresta **(1,2)**, resultam as duas subárvores:

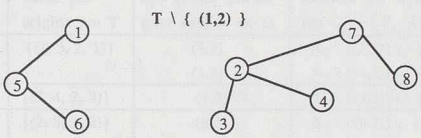


Fig. 6.8

A aresta que liga estas duas subárvores de menor custo é **(1,3)**, dado que:

$$c_{1,3} = 3 = \text{Min } \{ c_{1,3};\ c_{1,8};\ c_{5,8};\ c_{6,3} \}$$

Assim, uma *SST* em $G'_{(1,2)} = (V,\ A\backslash\{(1,2)\})$

é $T'_{(1,2)} = (V,\ A_T\backslash\{(1,2)\}\cup\{(1,3)\})$.

Por outro lado, **(1,3)** é também a aresta que substitui **(2,3)** numa *SST* em $G'_{(1,3)} = (V,\ A\backslash\{(2,3)\})$.

Analisando de outra perspectiva, a introdução em **T** da aresta de menor custo de **T"**, **(1,3)**, origina o ciclo **{(1, 3, 2, 1)}**. Assim, ela substitui as arestas $\mathbf{a_i}$ de **T** pertencentes a este ciclo, nas *SST's* que se pretendem obter nos grafos $G_i' = (V, A \backslash \{a_i\})$.

Se, no exemplo, forem ordenandas as arestas de **T"** por ordem crescente relativamente ao seu custo resulta:

$$\mathbf{A_T''} = \begin{bmatrix} (1,3) & (3,4) & (2,8) & (3,6) & (4,7) & (5,8) & (1,8) \\ 3 & 4 & 4 & 5 & 5 & 6 & 6 \end{bmatrix} \begin{matrix} \text{(aresta)} \\ \text{(custo)} \end{matrix}$$

Por este processo, as arestas das *SST's* nos **n-1** grafos G_i', são as de $\mathbf{A_T''}$, que figuram na Tab. 6.1.

Tab. 6.1.

Aresta de T"	Ciclo que origina em T	a_i - aresta que se quer excluir de G	Arestas de uma *SST* em $G_i' = (V, A \backslash \{a_i\})$
(1,3)	{(1, 3, 2, 1)}	(3,2)	$A_T \backslash \{(3,2)\} \cup \{(1,3)\}$
		(2,1)	$A_T \backslash \{(1,2)\} \cup \{(1,3)\}$
(3,4)	{(3, 4, 2, 3)}	(4,2) (*)	$A_T \backslash \{(4,2)\} \cup \{(3,4)\}$
(2,8)	{(2, 8, 7, 2)}	(8,7)	$A_T \backslash \{(8,7)\} \cup \{(2,8)\}$
		(7,2)	$A_T \backslash \{(2,7)\} \cup \{(2,8)\}$
(3,6)	{(3, 6, 5, 1, 2, 3)}	(6,5)	$A_T \backslash \{(6,5)\} \cup \{(3,6)\}$
		(5,1)	$A_T \backslash \{(5,1)\} \cup \{(3,6)\}$

(*) A aresta **(2,3)**, embora pertença ao ciclo originado pela introdução de **(3,4)** em **T**, já foi considerada quando da inclusão de **(1,3)** em **T**, que tem custo inferior ao de **(3,4)**. Logo, **(2,3)** é substituída por **(1,3)**.

Assim, as arestas de uma *SST* em **G"** são suficientes para se construirem as *SST's* em $G_i' = (V, A \backslash \{a_i\}), \forall\ a_i \in A_T$, as quais se podem determinar com base no algoritmo que se segue.

Algoritmo (A):

Dados: Matriz de custos, **C**; Uma *SST* em **G**, **T**;

Resultado: Uma *SST*, para cada um dos **n-1** grafos G_i'. Seja T_i', $i = 1,...,n-1$;

1. Construir **T"**, uma *SST* em **G"**;

2. Ordenar, por ordem crescente, relativamente aos custos, as arestas de **T"**;
Seja $\{b_1, b_2, ..., b_{n-1}\}$ o conjunto ordenado das arestas de **T"**;
$k \leftarrow 1$;

3. Identificar as arestas de **T** que pertencem ao ciclo originado pela introdução da aresta b_k em **T**. Para cada aresta a_i de **T** neste ciclo, uma *SST* em G_i' é:
$T_i' = (V, A_T \setminus \{a_i\} \cup \{b_k\})$;
$k \leftarrow k + 1$;

4. Parar se todas as arestas de **T** já foram analisadas;
C.c., identificar o ciclo formado pela introdução da aresta b_k em **T**. Para cada aresta não analisada, a_i de **T** neste ciclo, uma *SST* em G_i' é:
$T_i' = (V, A_T \setminus \{a_i\} \cup \{b_k\})$;
$k \leftarrow k + 1$;
Voltar ao início de **4**.

(b) Análise às arestas de uma *m-SST*

Um estudo idêntico para as arestas de uma *m-SST* não é tão simples dado ser necessário respeitar o grau do vértice **1**. Veja-se agora como abordar este problema. Para tal retome-se a formulação da *m-SST* apresentada no capítulo 5, aqui reproduzida por conveniência:

$$\text{Min} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \tilde{c}_{ij}\, x_{ij}$$

s.a:

$$\sum_{j \in S} x_{1j} = m \qquad (2)$$

$$\sum_{i,j \in S \atop i<j} x_{ij} \leq |S_k| - 1 \qquad \forall S_k \subseteq S,\ S_k \neq \emptyset \qquad (5)$$

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} x_{ij} = n-1 \qquad (8)$$

$$x_{ij} = 0,1 \qquad 1 \leq i < j \leq n \qquad (6)$$

Relaxando a restrição (2) que impõe que seja **m** o grau do vértice **1**, resulta:

$$\text{Min} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \tilde{c}_{ij}\, x_{ij} - w \left(\sum_{j \in S} x_{1j} - m \right)$$

s.a: (5), (6), (8)

o que é equivalente a:

$$wm + \text{Min} \sum_{j=2}^{n} \left(\sum_{i=1}^{j-1} \tilde{c}_{ij}\, x_{ij} - w\, x_{1j} \right)$$

s.a: (5), (6), (8)

Definindo:

$$d_{ij}(w) = \begin{cases} \tilde{c}_{ij} & \text{se } i \neq 1 \\ \tilde{c}_{ij} - w & \text{se } i = 1 \end{cases}$$

resulta o problema:

$$\textbf{(PR)} \quad wm + \text{Min} \sum_{j=2}^{n} \sum_{i=1}^{j-1} d_{ij}(w)\, x_{ij}$$

s.a: (5), (6), (8)

A solução óptima do problema relaxado **(PR)** é a solução óptima do problema inicial, se a restrição relaxada não fôr violada, i.é., se:

$$\sum_{j \in S} x_{1j} = m$$

Como se pode observar, **(PR)** traduz o problema da determinação de uma *SST* em **G**, onde a distância associada à aresta **(i,j)** é d_{ij}. Deste modo, se existir um valor de **w** para o qual **1** tenha grau **m** na *SST* solução de **(PR)**, está-se na presença da solução óptima do problema inicial.

No que se segue provar-se-á que se pode encontrar uma *m-SST* determinando uma *SST* num grafo em que se penalizem convenientemente as arestas incidentes em **1**. Torna-se então necessário demonstrar a existência de tal valor de **w**.

Sejam:

$$E = \{ e_1, e_2, ..., e_{n-2} \} = \{ (i_1, j_1), (i_2, j_2), ..., (i_{n-2}, j_{n-2}) \}$$

$$R = \{ (1, r_0), (1, r_1), ..., (1, r_{n-2}) \}$$

onde e_k ; $(1, r_k)$ é a troca de arestas que permite obter uma *(k+1)-SST* a partir de uma *k-SST*. Ou seja, se **T** fôr uma *k-SST*, $T \setminus \{e_k\} \cup \{(1,r_k)\}$ é uma *(k+1)-SST* (a existência de arestas nestas condições foi demonstrada no cap. 3 - teor.1);

$$W = \{ w_1, w_2, ..., w_{n-2} \}$$

Nas propriedades seguintes exprime-se a relação existente entre os valores de $w_i \in W$ e o multiplicador de Lagrange **w**.

Provar-se-à que, definindo: $w_k = \tilde{c}_{1,r_k} - \tilde{c}_{i_k,j_k}$, $k = 1, ..., n-2$

o resultado de **(PR)** para:

$w \leq w_1$ é uma *1-SST*;

$\mathbf{w} \in [\, w_{\ell-1}, w_{\ell} \,]$ é uma *ℓ-SST*;

$\mathbf{w} \geq w_{n-2}$ é uma *(n-1)-SST*.

Tendo em consideração que a diminuição do valor de **w** origina um aumento na distância d_{1j} ($= \tilde{c}_{1j} - w$), é óbvio que se **w** fôr muito pequeno a solução de **(PR)** é uma *1-SST*. Logo, a solução de **(PR)** para **w = -M** (em que **M** representa um número arbitràriamente grande) é uma *1-SST* e para **w = M** é uma *(n-1)-SST*. Assim, à medida que se diminui o valor de **w** está-se a penalizar o facto de se terem arestas incidentes em **1**.

Teor.1: Se existir um valor **w*** do multiplicador para o qual a solução de **(PR)** seja uma *k-SST*, então:

i) $\mathbf{w^*} \leq w_k$;

ii) Existe uma *(k+1)-SST* solução do **(PR)** para $\mathbf{w} = w_k$.

Dem.:

i) Suponha-se que $\mathbf{w^*} > w_k = \tilde{c}_{1,r_k} - \tilde{c}_{i_k,j_k}$

então,

$$d_{1,r_k}[w^*] - d_{i_k,j_k}[w^*] = \tilde{c}_{1,r_k} - w^* - \tilde{c}_{i_k,j_k}$$
$$< \tilde{c}_{1,r_k} - \tilde{c}_{1,r_k} + \tilde{c}_{i_k,j_k} - \tilde{c}_{i_k,j_k}$$
$$< 0$$

o que traduz que, se na solução de **(PR)** se substituir a aresta (i_k, j_k) por $(1, r_k)$, se obtém outra *SST* de custo inferior ao da actual. Assim, a solução de **(PR)**, para $\mathbf{w} = \mathbf{w^*}$, não era uma *k-SST*, o que contradiz a hipótese;

$\therefore \; \mathbf{w^*} \leq w_k$

ii) Seja, $w' = \tilde{c}_{1,r_k} - \tilde{c}_{i_k,j_k}$ (1)

Por definição, $d_{1,r_k}[w'] = \tilde{c}_{1,r_k} - w'$ (2)

(1) e (2) $\Rightarrow d_{1,r_k}[w'] = \tilde{c}_{i_k,j_k} = d_{i_k,j_k}[w']$

Logo, a árvore de suporte, que se obtém se à solução de **(PR)**, para $w=w'$, fôr introduzida a aresta $(1, r_k)$ e retirada (i_k, j_k), tem custo total igual ao da inicial;

Assim, se $w^* = w' = \tilde{c}_{1,r_k} - \tilde{c}_{i_k j_k}$, pode passar-se de uma *SST* solução de **(PR)** onde o vértice **1** tenha grau **k**, a uma *SST* de igual custo onde o vértice **1** tenha grau **k+1**. (c.q.d.)

Cor.1: Se um dado valor w' fôr tal que: $w_{k-1} \leq w' \leq w_k$, então, existe pelo menos uma *SST* solução de **(PR)**, para $w = w'$, que é uma *k-SST*.

Sejam:

T_k^- a *k-SST* solução de **(PR)**, para $w = w_k$;

T_k^+ uma solução de **(PR)**, para $w = w_k$, definida por:

$T_k^+ = T_k^- \setminus \{e_k\} \cup \{(1, r_k)\}$;

Logo, em T_k^+, **1** tem grau **k+1** e $T_k^- = T_{k-1}^+$;

$\{ a_1, a_2, ..., a_{n-1} \}$ as arestas de T_{k-1}^+;

Lema 1: Para cada aresta $a_i \in T_{k-1}^+$

i) existe uma árvore solução de **(PR)**, para $w = w_{k-1}$, no grafo $G_i' = (V, A\setminus\{a_i\})$, onde: $k-1 \leq grau(1) \leq k+1$;

ii) existe uma árvore solução de **(PR)**, para $w = w_{k-1}$, no grafo $G_i' = (V, A\setminus\{a_i\})$, onde: $k-2 \leq grau(1) \leq k$.

Dem.:

i) Seja T_1 a solução de (PR), para $w = w_{k-1}$, em G_i', que difere de T_{k-1}^+ numa só aresta (*). Ou seja: $T_1 = T_{k-1}^+ \setminus \{a_i\} \cup \{b_j\}$;

Podem então ter-se os seguintes casos:

1) a_i e b_j são ambas incidentes em **1** $\Rightarrow$ em T_1, **1** tem grau **k**;
2) se nenhuma das arestas incide em **1** $\Rightarrow$ em T_1, **1** tem grau **k**;
3) a_i incide em **1** e b_j não $\Rightarrow$ em T_1, **1** tem grau **k-1**;
4) a_i não incide em **1** e b_j incide $\Rightarrow$ em T_1, **1** tem grau **k+1**;

ii) Considere-se a árvore T_{k-1}^-. Pode ter-se um dos dois casos seguintes:

1) $a_i \notin T_{k-1}^-$
Seja T_2 a solução de (PR), para $w = w_{k-1}$, em G_i'
$\therefore$ $T_2 = T_{k-1}^- \Rightarrow$ em T_2, **grau(1) = k-1**;

2) Se $a_i \in T_{k-1}^-$;
Seja T_2 a solução de (PR), para $w = w_{k-1}$, em G_i' que difere de T_{k-1}^- por uma só aresta. Ou seja, $T_2 = T_{k-1}^- \setminus \{a_i\} \cup \{b_j\}$;

Se a_i incide em **1** e b_j não $\Rightarrow$ **grau(1) = k-2**, em T_2;
Se a_i não incide em **1** e b_j incide $\Rightarrow$ **grau(1) = k**, em T_2;
Se são ambas incidentes em **1** $\Rightarrow$ **grau(1) = k-1**, em T_2;
Se são ambas não incidentes em **1** $\Rightarrow$ **grau(1) = k-1**, em T_2; (c.q.d.)

As propriedades anteriores fundamentam o teorema seguinte, que serve de base ao algoritmo (V), utilizado na análise sensitiva às arestas da *m-SST*.

Teor. 2: Seja **T** uma *k-SST* em **G**. Então, para cada aresta $a_i \in T$, existe uma *k-SST*, T_i', em $G_i' = (V, A \setminus \{a_i\})$ tal que:

i) T_i' difere de **T** em duas arestas no máximo.

(*) Nota: Esta assumpção é válida, pois viu-se que uma *SST* em G_i' difere de uma *SST* em **G** por uma só aresta.

ii) Seja **T"** uma árvore solução de **(PR)**, para $w = w_{k-1}$, em $G'' = (V, A \setminus \{a_1, a_2, ..., a_n\})$;
As arestas de troca, ou seja que subtituêm a_i numa *k-SST* em G_i', pertencem a $A_T'' \cup \{e_{k-1}\} \cup \{(1, r_k)\}$.

Algoritmo (V) : (Análise Sensitiva às arestas da *m-SST*)

Dados: Matriz de custos transformada, $\tilde{C}$; Número de caixeiros, **m**;
Uma *m-SST*, **T**, em **G**, e o respectivo custo, **csst** ;

Resultado: Uma *m-SST*, T_i', para cada um dos **n-1** grafos G_i' e o respectivo custo, $csst_i$;

1. Identificar as arestas que se têm de trocar para se passar da *m-SST* dada a uma *(m-1)-SST* e a uma *(m+1)-SST*;

Sejam, respectivamente, $e_{m-1} = (i_{m-1}, j_{m-1}), (1, r_{m-1})$
e $e_m = (i_m, j_m), (1, r_m)$;

Calcular:

$w_{m-1} = \tilde{c}_{1,r_{m-1}} - \tilde{c}_{i_{m-1},j_{m-1}}$;

$w_m = \tilde{c}_{1,r_m} - \tilde{c}_{i_m,j_m}$;

2. Determinar uma *m-SST* em $G'' = (V, A'')$ (Algoritmo (I) - cap. 3);
Seja **T"** a *m-SST* e $A_T'' = \{b_1, b_2, ..., b_{n-1}\}$ o conjunto ordenado das suas arestas;

3. Encontrar todas as arestas de troca para as arestas da *m-SST* inicial (Algoritmo (A), com $\tilde{C}$ e **T**); Sejam $\{b_1, b_2, ..., b_s\}$;
$i \leftarrow 1$;

4. Para $a_i \in T$, seja b_j a aresta de troca;

4₁. Se a_i e b_j incidem ambas / nenhuma incide em **1**
a nova *m-SST* é $T_i' = T \setminus \{a_i\} \cup \{b_j\}$;
sendo o seu custo: $csst_i = csst - c_{a_i} + c_{b_j}$;

4₂. Se b_j é incidente em **1** e a_i não

Se $T \cup \{ e_{m-1} \} \setminus \{ a_i \}$ é uma árvore de suporte

$T_i' = T \cup \{ e_{m-1} \} \setminus \{ a_i \}$;

$csst_i = csst - c_{a_i} + c_{b_j}$;

C.c. $T_i' = T \cup \{e_{m-1}\} \cup \{b_j\} \setminus \{a_i\} \setminus \{(1, r_{m-1})\}$;

$csst_i = csst - c_{a_i} + c_{b_j}$;

4₃. Se a_i é incidente em **1** e b_j não

Se $T \cup \{(1, r_m)\} \setminus \{ a_i \}$ é uma árvore de suporte

$T_i' = T \cup \{(1, r_m)\} \setminus \{ a_i \}$;

$csst_i = csst - c_{a_i} + c_{b_j}$;

C.c. $T_i' = T \cup \{(1, r_m)\} \cup \{b_j\} \setminus \{a_j\} \setminus \{e_m\}$;

$csst_i = csst - c_{a_i} + c_{b_j}$;

5. $i \leftarrow i + 1$;

Se $i \leq n-1$ voltar a **4**;

C.c. terminar, foram analisadas todas as arestas da *m-SST*.

Torna-se assim possível saber o custo de uma *m-SST* para cada um dos grafos G_i'. Com base neste e no da *m-SST* inicial determina-se a penalidade de retirar a aresta a_i da solução em causa e o valor do novo problema relaxado. Se este exceder o majorante a_i é fixada com valor um.

6.3.2. Análise Sensitiva às Variáveis de Retorno

Como se viu, as arestas de retorno são as **m** de menor custo incidentes em **1**, caso estas conjuntamente com as da *m-SST* originem menos de **m** subcircuitos imediatos; ou as **m-1** de menor peso incidentes em **1** conjuntamente com a **m+1**-ésima de menor custo incidente em **1**, no caso contrário.

É óbvio que se fôr retirada uma das arestas de retorno a que a substitui, no primeiro caso em que foram detectados menos de **m** subcircuitos imediatos, é a **m+1**-ésima de menor custo incidente em **1**. No segundo caso, se a aresta que se retirar fôr a **m+1**-ésima de menor custo incidente em **1**, a que a substitui é a **m+2**-ésima nas mesmas condições, se a aresta retirada fôr outra a que a substitui é a que foi trocada pela **m+1**-ésima, ou seja a **m**-ésima de menor custo associado. Assim, a nova solução pode sempre encontrar-se com a troca de um par de arestas. Designe-se por $\hat{a}$ a aresta que é incluída na solução se fôr retirada uma aresta de retorno.

A análise de cada uma das arestas de retorno que não pertençam simultaneamente à *m-SST*, foi, então, elaborada calculando o valor do problema relaxado originado pela troca de arestas referida. Se este valor fôr maior que o valor do majorante a aresta em causa pode ser fixada com valor um.

As arestas de retorno que pertençam simultâneamente à *m-SST* têm que ser analisadas quando as da *m-SST*, pois o facto de serem retiradas origina por um lado uma troca de arestas como a que se acabou de frizar, e por outro alterações na *m-SST*. Tal foi feito alterando o método descrito para a análise às arestas da *m-SST* (Algoritmo (V)) nos seguintes passos:

4$_1$. Se a_i e b_j incidem ambas em **1**
a nova solução é $T_i' = T \setminus \{a_i\} \setminus \{a_j\} \cup \{b_j\} \cup \{\hat{a}\}$;
sendo o seu custo: $csst_i = csst - 2\, c_{a_i} + c_{b_j} + c_{\hat{a}}$;

4$_3$. Se a_i é incidente em **1** e b_j não
Se $T_1 = T \cup \{(1, r_m)\} \setminus \{a_i\}$ é uma árvore de suporte
a nova solução é $T_1 \setminus \{a_i\} \cup \{\hat{a}\}$;
C.c. a nova solução é $T_1 \setminus \{e_m\} \cup \{b_j\} \setminus \{a_i\} \cup \{\hat{a}\}$;

6.3.3. Análise Sensitiva às Restantes Variáveis

Neste ponto justifica-se a forma como foram fixadas, sempre que possível, variáveis que sendo livres não pertençam à solução do problema relaxado. Uma variável nestas condições, pode ser fixada com valor zero se o valor do minorante resultante da sua introdução na solução do problema relaxado exceder o valor do majorante.

A introdução duma destas arestas, na solução do problema relaxado, pode ou não originar uma alteração total da solução. Começa-se por descrever os casos em que a análise se pode fazer com a troca de apenas um par de arestas, finalizando-se com o estudo do caso mais complicado.

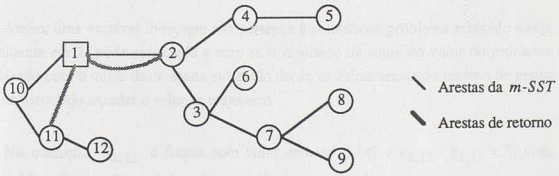
As arestas em análise podem ser de dois tipos:

(i) Incidentes em **1**;

(ii) Não incidentes em **1**.

Para mais fácil compreensão foi considerado o exemplo seguinte que acompanhará este estudo.

Ex.: Seja **n = 12** ; **m = 2** e



a solução do problema relaxado

Fig. 6.9

(i) Arestas incidentes em 1 :

O valor da solução do problema relaxado incluindo a aresta **(1,x)**, pode calcular-se com base na solução que se obtem substituindo a aresta de retorno de maior custo associado por **(1,x)**. Esta troca de arestas nunca origina mais subcircuitos imediatos que os existentes na solução do problema relaxado, pois **(1,x)** não pertencendo à solução não representa uma aresta da *m-SST*. Assim, a nova solução contem as **n-1** arestas da *m-SST*, as **m-1** arestas de menor custo incidentes em **1** e **(1,x)**.

Considerando-se o exemplo da Fig. 6.9, suponha-se que se pretendia analisar a aresta **(1,12)** e que $c_{1,2} < c_{1,11}$. Neste caso, a nova solução inclui **(1,12)** em vez de **(1,11)**, vindo:

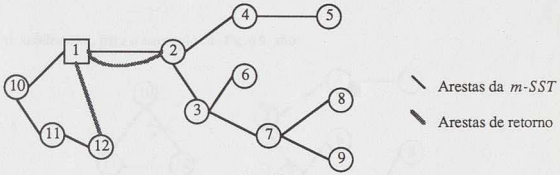


Fig. 6.10

Se **(1,2)** e **(1,10)** fossem as duas arestas de menor peso incidentes em **1** (caso em que foram detectados **m** subcircuitos imediatos, sendo **(1,10)** substituída por **(1,11)** na solução do problema relaxado), a nova solução viria igual à da Fig. 6.10, pois de qualquer forma inclui sempre as **m-1** arestas de menor custo incidentes em **1** e a aresta **(1,12)**.

Assim, uma variável livre, que não pertença à solução do problema relaxado e seja incidente em **1**, pode ser fixada a zero se o resultado da soma do valor do problema relaxado com o custo desta aresta subtraído do da **m**-ésima aresta de retorno de maior custo associado exceder o valor do majorante.

No exemplo, $x_{1,12}$ é fixada com valor zero se $L(\pi) + c_{1,12} - c_{1,11} > \overline{z}$, caso contrário, não se pode concluir nada e a variável permanece livre.

(ii) Arestas não incidentes em 1 :

Para se conhecer o valor do problema relaxado se a aresta (x,y), que não pertença a esta solução nem seja incidente em **1**, fôr introduzida na referida solução é necessário determinar a nova solução.

A introdução de **(x,y)** na solução origina um ciclo na *m-SST* e, como se vê de seguida, existem casos em que a nova *m-SST* pode ser encontrada se retirada uma das arestas desse ciclo.

Designe-se por *subárvores* as árvores que resultam de retirar o vértice **1** da *m-SST* e por *raíz* de uma *subárvore* o vértice dessa *subárvore* adjacente a **1** na *m-SST*.

As *subárvores*, para o exemplo da Fig. 6.9, são:

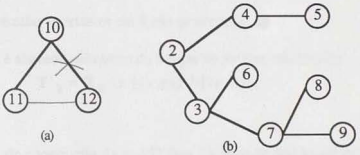


Fig. 6.11

Onde, **10** é a *raíz* da *subárvore* (a) e **2** a *raíz* de (b).

A aresta **(x,y)** pode estar num dos casos:

(ii_1) **x** e **y** pertencem à mesma *subárvore* ;

(ii_2) **x** e **y** pertencem a *subárvores* diferentes;

(ii_1) Sabe-se que uma *m-SST* é formada por **m** *SST's* em subgrafos gerados por conjuntos de vértices disjuntos e cuja união representa o conjunto **S**, pelo vértice **1** e por **m** arestas ligando **1** com cada uma das **m** *SST's*.

No que se segue provar-se-à que a *m-SST* que contém a aresta **(x,y)** é formada por **m** *SST's* nos mesmos subgrafos que a inicial. Assim, a nova *m-SST* pode ser encontrada por alteração de apenas uma das **m** *SST's*.

Seja:

$\mathbf{T = T_1 \cup ... \cup T_k \cup ... \cup T_m \cup \{(1,i_1)\} \cup ... \cup \{(1,i_k)\} \cup ... \cup \{(1,i_m)\}}$

a *m-SST* inicial;

$\mathbf{x, y \in T_k}$ com $\mathbf{(x,y) \notin T_k}$;

A introdução da aresta **(x,y)** em **T** origina um ciclo em $\mathbf{T_k}$. Provar-se-á de seguida que:

1º) As *SST's* $\mathbf{T_j ; j = 1 ,..., m, j \neq k}$ não se alteram;

2º) As arestas incidentes em **1** não se alteram;

3º) $\mathbf{T_k}$ é alterada pela troca de um par de arestas, resultando:

$$\mathbf{T'_k = T_k \cup \{(x,y)\} \setminus \{(a,b)\}};$$

1º) Pela forma de construção da *m-SST* (cap. 3), sabe-se que as arestas das *SST's*, $\mathbf{T_j}$ $\mathbf{j \neq k}$, foram escolhidas de acordo com as prioridades. A introdução de **(x,y)** em **T** não altera as prioridades das arestas destas árvores, não originando, portanto alterações nestas *SST's*, caso não haja alterações nos subgrafos. No fim vê-se que os subgrafos não se alteram.

2º) Pela razão anterior as arestas $\mathbf{(1,i_r)}$ com $\mathbf{r \neq k}$ não se alteram. A aresta $\mathbf{(1,i_k)}$ só seria alterada se:

$$\mathbf{c_{x,y} - c_{1,x} > c_{x,y} - c_{1,i_k} \Leftrightarrow c_{1,x} < c_{1,i_k} \quad i_k \neq x}$$

ou

$$\mathbf{c_{x,y} - c_{1,y} > c_{x,y} - c_{1,i_k} \Leftrightarrow c_{1,y} < c_{1,i_k} \quad i_k \neq y}$$

ou

$$\mathbf{c_{x,y} - c_{1,w} > c_{x,y} - c_{1,i_k} \Leftrightarrow c_{1,w} < c_{1,i_k} \quad i_k \neq w}$$

Mas, se tal fosse válido não existiria em **T** a aresta $(1,i_k)$ mas sim **(1,x)** ou (1,y) ou **(1,w)**, consoante o caso que se verificasse. Logo, não há alteração das arestas incidentes em **1**.

3º) Sendo T_k uma *SST* é óbvio que a introdução de **(x,y)** em T_k origina um e um só ciclo. Uma *SST* no mesmo subgrafo, que contenha **(x,y)**, resulta de substituir a aresta de maior custo associado, **(a,b)**, nesse ciclo por **(x,y)**.

$$\therefore \quad T'_k = T_k \cup \{(x,y)\} \setminus \{(a,b)\}$$ é uma *SST*, que contém **(x,y)**, no subgrafo gerado pelo mesmo conjunto de vértices.

Assim, se se provar que os subgrafos não se alteram, fica demonstrado que a *m-SST*, **T'**, que contém **(x,y)** difere de **T** num par de arestas, ou seja:

$$T'_k = T_k \cup \{(x,y)\} \setminus \{(a,b)\}$$

onde **(a,b)** representa a aresta de maior custo associado no ciclo originado pela introdução de **(x,y)** em **T**.

Os subgrafos só se alterariam se a introdução de **(x,y)** alterasse as prioridades das arestas que em T_k unem vértices de $T_k \setminus \{(a,b)\}$, podendo um destes vértices mudar de *SST*. Porém, a alteração da árvore, não tem nada a ver com estas filas de prioridades, como se ilustra no exemplo que se apresenta de seguida.

A nova *m-SST* é então formada por **m** *SST's* nos mesmos subgrafos que a inicial, o vértice **1** e as mesmas arestas a uni-lo com as **m** *SST's*. A única *SST* que virá alterada será a que contém os vértices **x** e **y**, sendo incluída a aresta **(x,y)** em substituição da aresta de maior custo associado no ciclo originado pela introdução de **(x,y)** nesta *SST*.

A aresta **(x,y)** é fixada com valor zero se o valor da nova solução do problema relaxado exceder o do majorante.

Ex.: Considere-se a introdução da aresta **(3,4)** no exemplo da Fig. 6.9. Os seus vértices pertencem à mesma subárvore (Fig. 6.11 (b)), sendo esta uma *SST* no subgrafo gerado pelo conjunto de vértices **{2, 3 ,..., 9}**.

Veja-se que a *m-SST* resultante da introdução de **(3,4)** na solução é formada pelas **m** *SST's* nos mesmos subgrafos que a inicial.

Suponha-se que $c_{2,3} > c_{2,4}$. É óbvio que uma *SST*, neste subgrafo, que inclua a aresta **(3,4)** é:

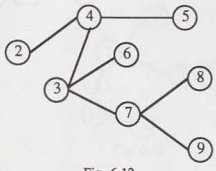


Fig. 6.12

Por razões óbvias, as únicas arestas que pertenciam à *m-SST* inicial e que poderiam não pertencer à nova são: **(3,6), (3,7), (7,8), (7,9)**. Porém, elas estão na *m-SST* inicial por representarem, a dado passo, a melhor troca de arestas. Por exemplo, **(3,6)** pertence à *m-SST* inicial porque a dado passo de construção desta árvore **(3,6) , (1,d)**, em que **d** é a raiz da árvore a que **3** ou **6** pertence, representou a troca de arestas prioritária, ou seja $c_{3,6} - c_{1,d}$ era mínimo. Esta relação não sendo alterada com a substituição de **(2,3)** por **(3,4)** mantem-se válida. Logo os subgrafos não se alteram.

(ii_2) Se **x** e **y** pertencem a *subárvores* diferentes o ciclo originado na *m-SST* com a introdução de **(x,y)** contém duas arestas incidentes em **1**. Assim, um procedimento semelhante ao efectuado em **(ii_1)** pode fazer com que se deixe de ter **grau(1)=m**, se fôr incidente em **1** a aresta de mair custo no ciclo. Por outro lado, mesmo sendo não incidente em **1** esta aresta, não se consegue garantir que a árvore resultante da substituição desta por **(x,y)** seja uma *m-SST*, como se mostra no exemplo seguinte.

Ex.: Considere-se, de novo o exemplo da Fig. 6.9. A inclusão da aresta **(3,11)** origina o ciclo **{(1, 10, 11, 3, 2, 1)}**. Suponha-se que:

$$c_{10,11} = \text{Máx} \{ c_{1,10}, c_{10,11}, c_{2,3}, c_{1,2} \}$$

e construa-se a nova árvore:

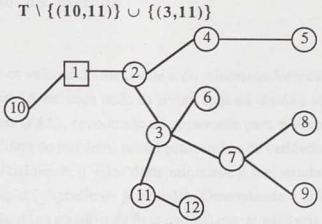


Fig. 6.13

Esta árvore pode não representar uma *m-SST*. Do que se sabe, a aresta **(11,12)** pertence à *m-SST* inicial porque $c_{11,12} - c_{1,11} \leq c_{11,12} - c_{1,10}$ e, para a árvore da Fig. 6.13 ser uma *m-SST* é necessário garantir que $c_{11,12} - c_{1,11} \leq c_{11,12} - c_{1,2}$. Tal seria válido se $c_{1,10} \geq c_{1,2}$. Podendo ainda ter-se uma solução com mais vértices ligados a **12**, a análise deste caso pode tornar-se bastante mais complicada, pois uma árvore em que **12**, ou qualquer um dos que estivessem ligados a **12**, fosse adjacente a **1**, pode ter custo inferior a esta.

Assim, para os casos em que **x** e **y** pertencem a *subárvores* diferentes foi encontrada uma nova *m-SST*, de acordo com o algoritmo (I) - cap. 3. Com base nesta nova árvore e nas arestas de retorno, determina-se a nova solução do problema relaxado. A aresta **(x,y)** é fixada com valor zero se o valor desta solução exceder o valor do majorante. As arestas de retorno só são alteradas se, conjuntamente com a nova *m-SST* originarem **m** subcircuitos imediatos.

6.4. Descrição do Método

Começando com os valores do majorante e do minorante fornecidos pelos métodos anteriormente descritos é, em cada nodo da árvore após escolhida a variável a fixar bem como o ramo a seguir (§ 6.2.), encontrado um minorante para o subproblema em causa (cada subproblema difere do problema inicial pelo número de variáveis já fixadas a zero e a um). Tal como inicialmente, o valor deste minorante é melhorado com o recurso ao método de óptimização subgradiente já descrito. Determinada a solução do problema relaxado, esta é analisada na tentativa de fixar implicitamente variáveis (§ 6.3.).

Com o objectivo de tentar reduzir o número de subproblemas a analisar, foi introduzido o seguinte critério de paragem:

$$(\overline{z} - \underline{z}) / \underline{z} \leq \varepsilon$$ onde: $\overline{z}$ é o valor do majorante
$\underline{z}$ é o valor do problema relaxado
ε é uma constante "pequena"

Assim, a diferença entre os valores da solução admissível, $\overline{z}$, que se obtem e o óptimo do *m-TSP* nunca excede ε.

Algoritmo (VI): (Determinação da solução óptima do *m-TSP*)

Dados: Matriz de custos, **C**; nº de cidades, **n**; nº de caixeiros viajantes, **m**; ε;

Resultado: Vector contendo as arestas da solução óptima, **SOL**;
Valor da solução óptima, $\overline{z}$;

1. Calcular uma solução admissível e o seu valor (Algoritmo (II) - cap. 4);
Seja **SOL** a solução e $\overline{z}$ o seu valor;

2. Resolver o problema Dual Lagrangeano (Algoritmo (IV) - cap. 5);
Seja, $\underline{z}$ o valor do minorante;

3. Parar se $(\overline{z} - \underline{z}) / \underline{z} \leq \varepsilon$ - **SOL** é uma solução óptima;
c.c., continuar;

4. Inicializações: **n_nodos** ← **1**; {nº de nodos na árvore}
niv_ram ← **0**; {nível da ramificação}
n_fixas1 ← **0**; {nº de variáveis fixadas a um}
n_fixas0 ← **0**; {nº de variáveis fixadas a zero}

5. Escolher a variável a fixar bem como o seu valor (§ 6.2.1.), seja x_{ij} ;
n_nodos ← **n_nodos + 1;**
niv_ram ← **niv_ram + 1;**

Se $x_{ij} = 1$ fazer:

$c_{ij} = c_{ji} \leftarrow -\infty$;
n_fixas1 ← **n_fixas1 + 1;**
Se **i (j)** ≠ **1** e se o nº de arestas fixadas a um e incidentes em **i (j)** fôr dois, fixar as restantes incidentes em **i (j)** a zero, actualizando **n_fixas0**;
C.c. se o nº de arestas fixadas a um e incidentes em **1** fôr **2m**, fixar as restantes arestas incidentes em **1** a zero, actualizando **n_fixas0**;

Se $x_{ij} = 0$ fazer:

$c_{ij} = c_{ji} \leftarrow \infty$;
n_fixas0 ← **n_fixas0 + 1;**
Se **i (j)** ≠ **1** e se o nº de arestas fixadas a zero e incidentes em **i (j)** fôr **n-3**, fixar as restantes incidentes em **i (j)** a um, actualizando **n_fixas1**;
C.c. se o nº de arestas fixadas a zero e incidentes em **1** fôr **n-m-2**, fixar as restantes arestas incidentes em **1** a um, actualizando **n_fixas1**;

6. Resolver o problema relaxado para o subproblema e melhorar o valor do minorante recorrendo ao método de optimização subgradiente (Algoritmo (IV) (*)), seja $\underline{z}$ o valor deste minorante;

(*) Nota: A inicialização dos valores dos multiplicadores (passo 1 do algoritmo IV) só é feita quando se faz o retrocesso na árvore. Caso contrário é considerado o vector de multiplicadores que proporcionou melhores valores no nodo anterior.

7. Se **n_fixas1** < **n-1+m** e **n_fixas0** < **n** (n-1)/2 - (**n-1+m**) tentar fixar implicitamente variáveis fazendo a análise sensitiva à solução do problema relaxado (Algoritmo (V) - § 6.3.);
Actualizar **n_fixas0** e **n_fixas1**;

8. Se **n_fixas1** < **n-1+m** ; **n_fixas0** < **n** (n-1)/2 - (n-1+m) ;
$\left(\bar{z} - \underline{z} \right) / \underline{z} > \varepsilon$ e $\underline{z} < \bar{z}$ voltar a **5**;

c.c. ir para **9**;

9. Se **n_fixas1** = **n+m-1** encontrar o valor desta solução admissível;

Se **n_fixas0** = **n(n-1)/2-(n+m-1)** encontrar a solução admissível formada por todas as variáveis livres e fixadas a um e calcular o seu valor;

c.c. ir para **10**;

Se o valor da solução admissível fôr menor que o actual majorante, actualizar o majorante, fazendo:

$\bar{z} \leftarrow$ **valor desta solução** e guardar a solução, **SOL**;

10. {Retrocesso na árvore}

Repetir

Considerar livres todas as variáveis fixadas neste nodo, decrementando o número de variáveis fixadas a zero ou a um e recuperando o custo inicial associado a cada variável;
niv_ram ← **niv_ram - 1**;
Considerar o nodo "pai" do actual;

Até ser encontrado um nodo com apenas um descendente analisado para o qual: $\left(\bar{z} - \underline{z} \right) / \underline{z} > \varepsilon$;

Parar se não existe nenhum nodo nestas condições - **SOL** representa a solução óptima;

C.c., fazer: **n_nodos** ← **n_nodos + 1**;
niv_ram ← **niv_ram + 1**;
ir para **11**;

11. Se, no subproblema representado pelo descendente deste nodo já analisado:

(i) x_{rs} foi fixada com valor um, fixá-la a zero, fazendo:

n_fixas1 ← **n_fixas1 - 1;**
$x_{rs} \leftarrow 0$;
$c_{rs} = c_{sr} \leftarrow \infty$;
n_fixas0 ← **n_fixas0 + 1;**

Se **i** (j) ≠ **1** e se o nº de arestas fixadas a zero e incidentes em **i** (j) fôr **n-3**, fixar as restantes incidentes em **i** (j) a um, actualizando **n_fixas1**;

C.c. se o nº de arestas fixadas a zero e incidentes em **1** fôr **n-m-2**, fixar as restantes arestas incidentes em **1** a um, actualizando **n_fixas1**;

(ii) x_{rs} foi fixada com valor zero, fixá-la a um, fazendo:

n_fixas0 ← **n_fixas0 - 1;**
$x_{rs} \leftarrow 1$;
$c_{rs} = c_{sr} \leftarrow -\infty$;
n_fixas1 ← **n_fixas1 + 1;**

Se **i** (j) ≠ **1** e se o nº de arestas fixadas a um e incidentes em **i** (j) fôr dois, fixar as restantes incidentes em **i** (j) a zero, actualizando **n_fixas0**;

C.c. se o nº de arestas fixadas a um e incidentes em **1** fôr **2m**, fixar as restantes arestas incidentes em **1** a zero, actualizando **n_fixas0**;

Voltar a **6**.

Nota: A solução designada por solução óptima, **SOL**, é a solução óptima exacta se $\varepsilon = 0$ e ε-optimal se $\varepsilon > 0$.

Diversas vezes os algoritmos de b&b recorrem a processos que permitem, sem grande esforço computacional, determinar em cada nodo da árvore uma solução admissível para o subproblema em causa, diferindo pouco da solução do problema relaxado. Tal procedimento auxilia a que sejam necessários menos nodos na ramificação, pois se o valor de uma solução admissível exceder o do majorante para o *m-TSP* pode ser cancelada a ramificação a partir desse nodo. Contudo, no método implementado não é vantajoso, segundo se pensa, a utilização de tais procedimentos.

7. Análise dos Resultados Computacionais

Os métodos apresentados foram testados num Microcomputador PC-AT/386, sistema operativo MS-DOS 3.30. Os programas foram codificados em Pascal e compilados em Turbo Pascal 5.0. O programa para a determinação do emparelhamento perfeito de custo mínimo [6] (cap. 4) estando codificado em Fortran foi compilado em Profor.

As coordenadas dos pontos foram geradas aleatóriamente, sendo de seguida calculadas as distâncias Euclideanas entre os diversos pontos. *Gavish & Srikanth* [21] com o mesmo tipo de dados, testam problemas com um máximo de **100** vértices e **10** caixeiros num IBM 3082, tendo sido, neste trabalho, conseguidas as dimensões máximas de **30** vértices e **3** caixeiros.

Embora o compilador utilizado seja o melhor que se conhece para programas codificados em Pascal em microcomputadores, tornou-se difícil testar problemas com mais de **20** vértices. Em primeiro lugar apresentam-se os resultados obtidos para problemas em que foi possível determinar a solução óptima pelo método exacto implementado. Sempre que por falta de espaço de memória o programa não chegou ao fim do b&b os resultados foram obtidos por etapas. Nestes casos, se o valor do majorante foi actualizado, repetiu-se o b&b desde o início com este novo valor em vez do fornecido pelo método heurístico. Houve, contudo, problemas em que não foi possível a determinação da solução óptima. Por este motivo a análise dos resultados para os diversos problemas é feita separadamente. Primeiro são analisados os resultados dos problemas de menores dimensões, sendo por fim estudados os restantes.

As tabelas 7.1 e 7.1(a) contêm informação sobre o número de problemas gerados, sendo agrupados os que se geraram em quadrados de iguais dimensões, indicando em quantos deles foi necessário entrar em b&b e o número médio de nodos efectuados. Os

resultados de problemas de iguais dimensões, separados nas diversas linhas, foram obtidos em quadrados de dimensões sucessivamente maiores. É ainda apresentado o tempo médio de execução gasto no nodo inicial, ou seja na determinação do minorante e do majorante iniciais, e o tempo médio de ramificação. Os resultados destes mesmos problemas, em que se trabalha com valores médios sempre que são gerados de igual forma, são detalhados nas tabelas 7.2 e 7.3.

Dimensões	Nº de Problemas		Nº de nodos			TCPU (em s)	
n, m	Total	B&B	Mín	Méd.	Máx	inicial	B&B
8, 2	2	1	-	31	-	0.41	6.18
8, 2	5	0	-	-	-	0.38	-
8, 2	5	1	-	5	-	0.45	0.58
8, 2	5	2	3	6	9	0.86	2.90
8, 2	5	4	3	6	11	1.39	1.76
8, 2	5	1	-	4	-	0.68	0.71
8, 3	6	0	-	-	-	0.80	-
8, 3	5	1	-	11	-	1.09	6.66
8, 3	5	1	-	3	-	1.01	0.44
8, 3	5	3	7	16	21	1.31	10.85
8, 3	5	2	12	26	40	1.26	17.57
11, 2	5	1	-	13	-	1.55	10.36
11, 2	5	1	-	16	-	1.70	15.53
11, 2	5	2	7	10	12	2.64	9.28
11, 2	5	4	7	9	13	3.84	10.20
11, 2	5	5	3	18	49	4.84	16.97
11, 3	3	3	10	37	71	3.82	64.95
11, 3	3	0	-	-	-	1.83	-
11, 3	3	1	-	24	-	2.82	36.40
11, 3	3	1	-	11	-	2.37	19.84
11, 3	3	2	27	61	94	4.10	108.37
12, 2	5	3	9	13	18	1.54	7.52
12, 2	5	3	17	25	33	3.54	27.43
12, 2	5	4	3	24	48	2.83	19.17
12, 2	5	3	14	31	61	3.67	42.52
12, 2	4	3	5	8	14	4.43	9.21

Tab. 7.1

Dimensões	Nº de Problemas		Nº de nodos			TCPU (em s)	
n, m	Total	B&B	Mín	Méd.	Máx	inicial	B&B
12, 3	3	2	21	21	21	4.03	39.43
12, 3	3	1	-	14	-	3.06	31.47
12, 3	4	3	3	16	35	4.51	25.38
12, 3	3	2	13	22	30	4.52	43.05
12, 3	3	2	27	33	39	5.25	67.11
12, 4	1	1	-	19	-	2.91	17.71
12, 4	1	0	-	-	-	1.12	-
12, 4	1	0	-	-	-	1.16	-
12, 4	1	0	-	-	-	1.44	-
12, 4	1	1	-	41	-	2.84	20.71
15, 2	5	2	27	49	71	1.43	57.33
15, 2	3	2	26	30	34	6.34	66.42
15, 2	5	1	-	56	-	1.92	63.36
15, 2	3	2	7	22	37	6.11	56.38
15, 2	3	2	3	16	29	7.51	28.18
15, 2	5	0	-	-	-	4.62	-
15, 3	2	0	-	-	-	2.09	-
15, 3	3	0	-	-	-	3.93	-
15, 3	5	1	-	65	-	4.07	165.14
15, 3	3	2	39	59	79	7.81	210.77
15, 3	4	3	8	18	25	10.66	65.02
20, 2	3	3	29	50	83	19.84	291.15
20, 2	4	4	56	66	77	15.58	316.75
20, 2	5	2	65	123	181	11.88	672.04
20, 2	2	2	70	72	73	16.31	344.38
20, 2	3	3	25	63	119	31.94	298.79

Nota: TCPU representa o tempo de CPU gasto na execução dos programas.

Tab. 7.1 (a)

Desta tabela pode observar-se que em **94** dos **191** problemas teste foi necessário ramificar.

Não tendo sido implementadas regras alternativas de fixação explícita de variáveis, torna-se difícil comentar a regra utilizada. Pode-se porém afirmar não ser, na maior parte dos casos, elevado o número médio de nodos necessários para obter a solução óptima. Nos resultados obtidos observou-se que embora raras vezes a solução óptima coincidisse com a solução dos últimos subproblemas de cada árvore, também não era representada pela solução dos primeiros subproblemas que se conseguiam resolver. Tal leva a suspeitar que a regra adoptada talvez não seja a que proporcione melhores resultados.

Os desvios percentuais do majorante e do minorante que constam nas tabelas seguintes foram determinados, respectivamente, de acordo com as fórmulas:

$$(\overline{z} - z^*) / z^* \quad \text{e} \quad (z^* - \underline{z}) / z^*$$

onde : z^* é o valor da solução óptima;

$\overline{z}$ é o valor do majorante no nodo inicial;

$\underline{z}$ é o valor do minorante no nodo inicial.

Estes valores foram calculados para cada problema testado, sendo nas tabelas 7.2 e 7.3 apresentadas as médias dos problemas gerados de igual forma.

Dimensões	Desvio Percentual do		Dimensões	Desvio Percentual do	
n, m	majorante	minorante	n, m	majorante	minorante
8, 2	9.03%	2.56%			
8, 2	11.68%	0.47%	8, 3	0.23%	0.00%
8, 2	5.73%	2.32%	8, 3	14.41%	2.52%
8, 2	10.3%	0.72%	8, 3	19.17%	0.76%
8, 2	4.75%	1.40%	8, 3	31.12%	2.79%
8, 2	7.91%	0.44%	8, 3	18.34%	3.96%
11, 2	5.15%	1.23%	11, 3	9.26%	5.95%
11, 2	7.18%	0.35%	11, 3	19.93%	0.00%
11, 2	6.76%	1.84%	11, 3	18.77%	2.56%
11, 2	5.77%	3.32%	11, 3	18.24%	0.00%
11, 2	2.32%	2.40%	11, 3	4.87%	6.28%
12, 2	0.12%	0.00%	12, 3	8.51%	2.43%
12, 2	0.09%	0.02%	12, 3	22.33%	0.25%
12, 2	5.47%	3.62%	12, 3	8.49%	1.41%

Tab. 7.2

Dimensões	Desvio Percentual do		Dimensões	Desvio Percentual do	
n, m	majorante	minorante	n, m	majorante	minorante
12, 2	5.96%	3.63%	12, 3	11.75%	1.23%
12, 2	1.81%	1.82%	12, 3	31.86%	2.04%
12, 4	0.00%	7.22%	15, 2	13.57%	3.39%
12, 4	22.97%	0.00%	15, 2	8.83%	2.56%
12, 4	18.18%	0.00%	15, 2	11.27%	1.54%
12, 4	19.64%	0.00%	15, 2	11.63%	2.57%
12, 4	13.71%	9.68%	15, 2	8.47%	0.00%
15, 3	17.27%	0.29%	20, 2	7.22%	3.59%
15, 3	18.58%	0.00%	20, 2	10.83%	3.22%
15, 3	12.15%	0.00%	20, 2	7.60%	0.66%
15, 3	11.79%	2.29%	20, 2	13.37%	1.14%
15, 3	9.74%	5.47%	20, 2	0.00%	4.45%

Tab. 7.3

Como se pode observar nos resultados em análise o desvio percentual do majorante, relativamente ao valor óptimo do problema, é na maior parte dos casos bastante superior ao desvio percentual do minorante. Considere-se o quadro seguinte que resume alguns resultados das tabelas em foco.

Dimensões	Valor mínimo para o desvio do		Valor máximo para o desvio do	
n, m	Majorante	Minorante	Majorante	Minorante
8, 2	4.75%	0.44%	11.68%	2.56%
8, 3	0.23%	0.00%	31.12%	3.96%
11, 2	2.32%	0.35%	7.18%	3.32%
11, 3	4.97%	0.00%	19.93%	6.98%
12, 2	0.09%	0.00%	5.96%	3.63%
12, 3	8.49%	0.25%	31.86%	2.43%
12, 4	0.00%	0.00%	22.97%	9.68%
15, 2	3.25%	0.00%	13.57%	3.39%
15, 3	9.74%	0.00%	18.58%	5.47%
20, 2	0.00%	0.66%	13.37%	4.45%
Máx	9.47%	0.44%	31.86%	9.68%
Mín	0.00%	0.00%	5.96%	2.56%

Held et al. [28] afirmam, com base em testes que realizaram, que os resultados obtidos pelo método subgradiente não parecem depender muito do valor do majorante. Tal justifica que o método descrito para a determinação de minorantes, embora faça uso do majorante, conduza a resultados significativamente mais próximos do valor óptimo que os obtidos pela heurística implementada.

Deve ainda salientar-se que dos **97** problemas cuja solução óptima foi encontrada no nodo inicial apenas em **8** (cinco dos quais dizem respeito aos problemas de menores dimensões) se obteve a igualdade entre os valores do minorante e do majorante. Nos restantes **89** a solução óptima foi encontrada por ser admissível a solução do problema relaxado. Este facto verificou-se ainda diversas vezes na ramificação, obtendo-se soluções dos problemas relaxados que representaram soluções admissíveis dos subproblemas.

Gavish & Srikanth [21], não referindo qual o método heurístico que utilizam, propõem o uso de majorantes artificiais, sempre que a diferença entre o valor do majorante e o do minorante for relevante. Um majorante artificial é um valor menor que o valor da solução admissível fornecida pela heurística e superior ao melhor minorante. A escolha de tal majorante origina uma significativa redução no tempo de excução ([21]), se no método de ramificação em árvore for encontrada uma solução admissível de valor menor ou igual ao do majorante artificial. Caso tal não seja possível, significa que se tomou para majorante um valor inferior ao do valor óptimo do problema, tendo que se repetir o processo desde o início o que provoca um aumento significativo no tempo de execução computacional. Segundo afirmam, poucas vezes necessitaram de repetir o b&b e na tabela de resultados que apresentam, em **62** problemas, que necessitaram de ramificação, o processo foi repetido apenas duas vezes nos problemas de menores dimensões.

Os resultados das tabelas 7.4 e 7.5 conduzem a uma análise semelhante à já feita para os problemas de menores dimensões.

Note-se porém que nestes problemas o desvio médio do majorante, embora mantendo-se superior ao do minorante, diminui com o aumento das dimensões do problema.

Embora em proporções menores, também para estes problemas houve casos em que a solução do problema relaxado representou a solução óptima do problema.

Dimensões	Majorante	Minorante	Solução Óptima		Desvio Percentual do		TCPU (s)	
n, m	inicial	inicial	Valor	Nº nodos	majorante	minorante	inicial	b&b
20, 3	117 110 109	101	109	190 139 84	7.34%	7.34%	17.7	1 097.5 658.7 314.5
	169 166 164 154	149	149	45 55 71 36	13.42%	0.00%	7.1	148.4 175.1 168.4 70.6
	166 164	140	140	79 0	18.57%	0.00%	8.1	219.7
Média					**13.11%**	**2.45%**		
25, 2	74 73 72 70	65	66	54 64 103 138	12.12%	1.54%	24.0	385.7 480.4 673.1 774.0
	175	162	162	——	8.02%	0.00%	3.1	——
	221 206	199	206	37 23	7.30%	3.40%	27.2	265.8 119.8
Média					**10.14%**	**1.5%**		
25, 3	147	143	143	8	2.80%	0.00%	24.2	25.7
	189	157	157	——	20.38%	0.00%	5.9	——
	146	137	137	——	6.57%	0.00%	3.3	——
	159 151	147	151	76 19	5.30%	2.65%	15.6	399.8 47.5
Média					**8.76%**	**0.60%**		
30, 2	168	161	165	89	1.82%	2.42%	28.9	368.0
	78 77 73	71	72	102 110 28	8.33%	1.39%	24.2	667.4 703.2 135.0
	69 66 64	60	61	49 169 61	13.11%	1.64%	20.1	422.3 1 264.7 658.5
	188	169	169	——	11.24%	0.00%	5.6	——
	165 161 157	154	157	59 111 35	5.10%	1.91%	23.6	367.0 566.9 130.4
Média					**7.92%**	**1.47%**		

Tab. 7.4

Dimensões	Majorante	Minorante	Solução Óptima		Desvio Percentual do		TCPU (s)	
n, m	inicial	inicial	Valor	Nº nodos	majorante	minorante	inicial	b&b
30, 3	167	157	161	99	3.73%	2.48%	27.9	578.7
	161 154	149	154	70 39	4.55%	3.25%	27.6	479.7 162.2
	199	189	192	19	3.65%	1.56%	43.6	126.0
Média					**3.97%**	**2.43%**		

Tab. 7.5

Para finalizar este análise resta dizer que houve exemplos em que não foi possível encontrar a solução óptima do problema por o programa ter "rebentado" sem actualizar o valor do majorante, dado a problemas de falta de espaço de memória. Por não ser possível qualquer conclusão com base nestes casos os resultados não são apresentados.

Alguns autores comparam o método que apresentam com outros já existentes para a determinação da solução óptima do *m-TSP* e referidos ao longo deste trabalho. De seguida é feito um breve resumo destas comparações para problemas Euclideanos.

Gavish & Srikanth [21] comparam o seu método com os de *Laporte & Nobert* [34] e de *Ali & Kennington* [1], concluindo que resolvendo problemas de maiores dimensões é ainda, em termos de tempo de execução, comparável ao primeiro e mais rápido que o segundo.

Jonker & Volgenant [31] geram dados em rectângulos de dimensões diferentes que os gerados por *Gavish & Srikanth*, afirmando que os dados gerados por estes autores conduzem a melhores resultados comparativamente aos por eles gerados. Para problemas Euclideanos com um máximo de **80** cidades, obtêm geralmente minorantes menos afastados do valor óptimo do problema, necessitando ainda de menos nodos na ramificação. Afirmam ser de esperar que o seu método seja tanto melhor quanto maior fôr o número de caixeiros. O método de *Gavish & Srikanth*, como os próprios autores referem, não proporciona conclusões a este respeito.

8. Conclusões

Com base na análise dos resultados feita no capítulo anterior pode inferir-se que o método heurístico de *Frieze*, embora seja o que proporciona melhor análise de pior caso, em problemas simétricos e em que é válida a desigualdade triangular, caso em que se englobam os problemas testados, não conduz em geral a bons resultados práticos. Este resultado está em concordância com o obtido na implementação computacional da heurística de *Christofides* para o problema do caixeiro viajante simples ([39]). Trata-se portanto de um método com interesse essencialmente teórico, por ser o que tem melhor análise de pior caso sob condições de simetria e sendo válida a desigualdade triangular.

O método subgradiente utilizado origina significativas melhorias para os valores dos minorantes, proporcionando, como se viu, valores não muito afastados do valor óptimo do problema. Conforme anteriormente referido, a inicialização dos multiplicadores proposta por *Gavish & Srikanth* [21] foi testada, tendo-se observado originar melhores minorantes iniciais do que tomar como vector de multiplicadores inicial o vector nulo.

A regra de fixação implícita de variáveis não conduz "rápidamente" à obtenção da solução óptima, sendo de suspeitar que talvez uma das outras regras mencionadas (§ 6.2.) proporcione melhores resultados.

Sendo os valores dos minorantes "razoáveis" e os dos majorantes "afastados" do valor óptimo, não é descabido pensar que o uso de majorantes artificiais proporcione a resolução de problemas de maiores dimensões. Porém, tal não foi testado ficando em aberto para trabalhos futuros.

Uma forma de reduzir o tempo de execução e o espaço de memória necessário ao algoritmo implementado, seria fazer a análise sensitiva às variáveis que não fazem parte da solução do problema relaxado e que unem vértices de subárvores distintas (&6.3.3. - ($\overline{ii}_2$)) sem recalcular uma nova *m-SST*. Porém, tal só é possível se se provar a existência de um processo de reencontrar a nova *m-SST* a partir da que faz parte da solução do problema relaxado.

Sendo o *m-TSP* generalizável a problemas de maior aplicabilidade prática torna-se também importante a generalização dos métodos para ele existentes. *Gavish & Srikanth* [21] apresentam duas generalizações para o método proposto, que se descrevem de seguida.

Afirmam ser possível provar que um *m-TSP* em que **m** é variável pode ser resolvido pelo método que apresentam para o *m-TSP* com **m** fixo, num grafo com mais **m-1** vértices que o inicial, ou seja com **n+m-1** vértices.

O método pode ainda fácilmente generalizar-se a problemas em que existem limites para o número total de subcircuitos imediatos, bastando para tal alterar a forma como são encontradas **m** as arestas representativas do retorno dos caixeiros à cidade inicial.

Referências Bibliográficas

[1] **Ali, A.I.; Kennington, J.L.** - 1984
"The Asymmetric M-Travelling Salesmen Problem: A Duality Based Branch-and-Bound Algorithm"
Discrete Applied Mathematics, Vol. 13, pg. 259 - 276;

[2] **Bazaraa, M.S. ; Goode, J.J.** - 1977
"The Traveling Salesman Problem: A Duality Approach"
Mathematical Programming, Vol. 13, pg. 221 - 237;

[3] **Bellmore, M. ; Malone, J.** - 1971
"Pathology of Traveling-Salesman Subtour-Elimination Algorithms"
Operations Research, Vol. 19, pg. 278 - 307;

[4] **Bellmore, M. ; Hong, S.** - 1974
"Transformation of Multisalesmen Problem to the Standard Traveling Salesman Problem"
Journal of the Association for Computing Machinery, Vol. 21, pg. 500 - 504;

[5] **Berenguer, X.** - 1979
"A Characterization of Linear Admissible Transformations for the m-Travelling Salesmen Problem"
European Journal of Operational Research, Vol. 3, pg. 232 - 238;

[6] **Bukard, R.E. ; Derigs, U.** - 1980
"Assignment and Matchings Problems : Solution methods with Fortran-Programs"
Lecture Notes in Economics and Mathematical Systems (184) , Springer-Verlag;

[7] **Cheriton, D. ; Tarjan, R.E.** - 1976
"Finding Minimum Spanning Trees"
Siam Journal of Computing, Vol. 5 (4), pg. 724 - 742;

[8] **Chin, F. ; Houck, D.** - 1978
"Algorithms for Updating Spanning Trees"
Journal of Computer and System Sciences, Vol. 16, pg. 333 - 344;

[9] **Christofides, N.** - 1979
"The Travelling Salesman Problem"
in Combinatorial Optimization, edited by Christofides, N.; Mingozzi, A.; Toth, P.; Sandi, C.; John Whiley & Sons;

[10] **Christofides, N. ; Mingozzi, A. ; Toth, P.** - 1981
"Exact Algorithms for the Vehicle Routing Problem, Based on Spanning Tree and Shortest Path Relaxations"
Mathematical Programming, Vol. 20, pg. 283 - 302;

[11] **Dantzig, G.B. ; Fulkerson, D.R. ; Johnson, S.M.** - 1954
"Solution of a Large-Scale Traveling- Salesman Problem"
Operations Research, Vol. 2, pg. 293 - 410;

[12] **Dantzig, G.B. ; Fulkerson, D.R. ; Johnson, S.M.** - 1959
"On a Linear-Programming, Combinatorial Approach to the Traveling-Salesman Problem"
Operations Research, Vol. 7, pg. 58 - 66;

[13] Dijkstra, E.W. - 1959
"*A Note on Two Problems in Connexion with Graphs*"
Numerische Mathematik, Vol. 1, pg. 269 - 271;

[14] Discenza, J.H. - 1981
"*A More Compact Formulation of the Multiple Traveling Salesman Problem with Fixed*
Networks, Vol. 11, pg. 73 - 75;

[15] Eastman, W.L. - 1958
"*Linear Programming with Pattern Constraints*"
Ph. D. Thesis, Harvard University, Cambridge, MA;

[16] Frederickson, G.N. ; Hetch, M.S. ; Kim, C.E. - 1978
"*Approximation Algorithms for some Routing Problems*"
Siam Journal on Computing, Vol. 7, pg. 178 - 193;

[17] Frieze, A.M. - 1983
"*An Extension of Christofides Heuristic to the k-Person Travelling Salesman Problem*"
Discrete Applied Mathematics, Vol. 6, pg. 79 - 83;

[18] Gabow, H.N. - 1978
"*A Good Algorithm for Smallest Spanning Trees with a Degree Constraint*"
Networks, Vol. 8, pg. 201 - 208;

[19] Garey, M.R. ; Johnson, D.S. - 1979
Computers and Interactibility - A Guide to the Theory of NP-Completeness
W.H. Freeman & Co. - San Francisco;

[20] Gavish, B. ; Srikanth, K. - 1979
"*$O(n^2)$ Algorithms for Sensitivity Analysis of Minimal Spanning Trees and Related Subgraphs*"
Working Paper No. 8003, Graduate School of Management, University of Rochester;

[21] Gavish, B. ; Srikanth, K. - 1986
"*An Optimal Solution Method for Large-Scale Multiple Traveling Salesman Problems*"
Operations Research, Vol. 34 (5), pg. 698 - 717;

[22] Geofrion, A. - 1974
"*Lagrangean Relaxation for Integer Programming*"
Mathematical Programming Study, Vol. 2, pg. 82 - 114;

[23] Glover, F. ; Klingman, D. - 1974
"*Finding Minimum Spanning Trees with a Fixed Number of Links at a Node*"
in Combinatorial Programming: Methods and Applications
edited by B. Roy; D. Reidel Publishing Company; Dordretch-Holland/Boston - USA;

[24] Golden, B. ; Bodin, L. ; Doyle, T. ; W. Stewart, Jr. - 1980
"*Approximate Traveling Salesman Algorithms*"
Operations Research, Vol. 28, pg. 694 - 711;

[25] Gondran, M. ; Minoux, M. - 1984
Graphs and Algorithms
John Willy & Sons;

[26] Held, M. ; Karp, R.M. - 1970
"*The Traveling Salesman Problem and Minimum Spanning Trees*"
Operations Research, Vol. 18, pg. 1138 - 1162;

[27] Held, M. ; Karp, R.M. - 1971
"The Traveling Salesman Problem and Minimum Spanning Trees, Part 2"
Mathematical Programming, Vol. 1, pg. 6 - 25;

[28] Held, M. ; Wolfe, P.; Crowder, H. - 1974
"Validation of Subgradient Optimization"
Mathematical Programming, Vol. 6, pg. 62 - 88;

[29] Hong, S. ; Padberg, H.W. - 1977
"A Note on the Symmetric Multiple Traveling Salesman Problem with Fixed Charges"
Operations Research, Vol. 25, pg. 871 - 874;

[30] Hu, T.C. - 1974
"Optimum Communication Spanning Trees"
Siam Journal on Computing, Vol. 3 (3), pg. 188 - 195;

[31] Jonker, R. ; Volgenant, T. - 1982
"A Branch and Bound Algorithm for the Symetric Traveling Salesman Problem Based on the 1-Tree Relaxation"
European Journal of Operational Reserach, Vol. 9, pg. 83 - 89;

[32] Jonker, R. ; Volgenant, T. - 1988
"An Improved Transformation of the Symetric Multiple Traveling Salesman Problem"
Operations Research, Vol. 36 (1), pg. 163 - 167;

[33] Kruskal, J.B. - 1956
"On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem"
Proc. Amer. Math. Soc., Vol. 7, pg. 48 - 50;

[34] Laporte, G. ; Nobert, Y. - 1980
"A Cutting Planes Algorithm for the m-Salesman Problem"
Journal of the Operational Research Society, Vol. 31 (11), pg. 1017 - 1023;

[35] Lenstra, J.K. ; Rinnooy Kan, A.H.G. - 1976
"On General Routing Problems"
Networks, Vol. 6, pg. 273 - 280;

[36] Lenstra, J.K. ; Rinnooy Kan, A.H.G. - 1979
"A Characterization of Linear Admissible Transformations for the m-Travelling Salesmen Problem: A Ressult of Berenguer"
European Journal of Operational Research, Vol. 3, pg. 250 - 252;

[37] Little, J.D.C. ; Murty, K.G. ; Sweeney, D.W. ; Karel, C. - 1963
"An Algorithm for the Traveling Salesman Problem"
Operations Research, Vol. 11, pg. 972 - 989;

[38] Miller, C.E. ; Tucker, A.W. ; Zemlin, R.A. - 1960
"Integer Programming Formulation of Traveling Salesman Problems"
Journal of the Association for Computing Machinery, Vol. 7, pg. 326 - 329;

[39] Mourão, M.C. - 1988
"Heurística de Christofides para o Problema do Caixeiro Viajante - um estudo computacional"
Documento de trabalho nº 42, Cemapre, ISE;

[40] Orloff, C.S. - 1974
"Routing a Fleet of M Vehicles to/from a Central Facility"
Networks, Vol. 4, pg. 147 - 162;

[41] Prim, R.C. - 1957
"*Shortest Interconnection Networks and Some Generalizations*"
Bell System Technical Journal, Vol. 36, pg. 1389 - 1401;

[42] Rao, M.R. - 1980
"*A Note on the Multiple Traveling Salesman Problem*"
Operations Research, Vol. 28, pg. 628 - 632;

[43] Spira, P.M. ; Pan, A. - 1975
"*On Finding and Updating Spanning Trees and Shortest Paths*"
Siam Journal on Computing, Vol. 4 (3), pg. 357 - 380;

[44] Syslo, M.; Deo, N.; Kowalik, J.S. - 1983
Discrete Optimization Algorithms with Pascal Programs
Prentice Hall, Inc.;

[45] Yao, A.C. - 1975
"*An O(|E| log log |V|) Algorithm for Finding Minimum Spanning Trees*"
Inform. Proc. Lett., Vol. 4, pg. 21 - 23;