UMA EXTÊNSÃO DO PROBLEMA DE ROTEAMENTO DE VEÍCULOS: PROBLEMA DE ROTEAMENTO DE AUDITORES

MURILO CASTELLANO

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

Aprovada por:

Prof.Cláudio Thomas Bornstein, Dr.Rer.Nat.
(Presidente)

Prof.Roberto Diéguez Galvão, Ph.D.

Prof.Leonardo Junqueira Lustosa, Ph.D

RIO DE JANEIRO, RJ - BRASIL

CASTELLANO, MURILO

Uma extensão do Problema de Roteamento de Veículos: Problema de Roteamento de Auditores [Rio de Janeiro] 1990

V, 152 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas, 1990)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Otimização Combinatória I. COPPE/UFRJ II. Título (série)

AGRADECIMENTOS

Ao Dr.Clóvis Augusto Ribeiro, funcionário da direção geral do Banco do Brasil S.A., pelas sugestões e incentivo ao meu trabalho.

A Lenise, pela dedicação e compreensão nos momentos difíceis.

A Maria Cândida e Maria Cristina, pela sincera amizade e ajuda em todos os momentos.

Aos colegas, funcionários do Banco do Brasil S.A., Marco, Nelson e Nilo, pelo apoio e presteza no fornecimento dos dados imprescindíveis a este trabalho.

Em especial, ao meu orientador, Professor Cláudio Thomaz Bornstein, pelo apoio, compreensão e, sobretudo, sua valiosa orientação.

A minha família, que sempre deu sustentação a todos os meus empreendimentos.

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

UMA EXTENSÃO DO PROBLEMA DE ROTEAMENTO DE VEÍCULOS: PROBLEMA DE ROTEAMENTO DE AUDITORES

MURILO CASTELLANO

MARÇO/90

Orientador: Cláudio Thomas Bornstein
Programa : Engenharia de Sistemas

Estudamos o Problema de Roteamento dos Auditores do Banco do Brasil S.A., modelando-o de forma semelhante ao clássico Problema de roteamento de Veículos encontrado na literatura.

Desenvolvemos um algoritmo heurístico original, resultado da combinação de técnicas de Análise de Grupamentos com um procedimento sequencial do tipo "saving", capaz de resolver o Problema dos Auditores, bem como, os assemelhados Problema de Roteamento de Veículos, um e multi-depósito.

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

AN EXTENSION OF THE VEHICLE ROUTING PROBLEM: THE AUDITOR ROUTING PROBLEM

MURILO CASTELLANO

MARCH/90

Thesis supervisor: Cláudio Thomas Bornstein
Department : Sistems Program

We examine the routing of auditors of Banco do Brasil S.A., the Brazilian State Bank. This Problem is similar to the classical Vehicle Routing Problem. After presenting some models we develop an heuristic algorithm which results from the association of cluster analysis thechniques and a sequential saving procedure. This algorithm is able to solve The Auditor Problem as well as Single and Multiple Depot Vehicle Routing Problems. Computational results are presented.

Í N D I C E

CAPÍTULO I

INTRODUÇÃO

O Banco do Brasil S.A. possui uma rede de mais de quatro mil dependências situadas no país que recebem periódicamente a visita de auditores do seu próprio quadro de funcionários. Os auditores estão localizados nos mais diversos pontos do país, denominados praças-sedes, dos quais partem e regressam ao final de uma rota. Existe um critério interno que determina o tempo em que cada auditor deverá permanecer na dependência a fim de realizar a auditoria. Tal critério está relacionado com o porte da dependência e, também, com questões de segurança interna do Banco. Não se pode, portanto, determinar com muita antecedência, por exemplo um ano, o período certo em que uma dependência deverá ser visitada, uma vez que a determinação de tal período seria influenciada pela evolução dos serviços da dependência. É usual, portanto, considerar um período de planejamento menor, 60 a 70 dias, quando, então, seriam selecionadas aquelas dependências que necessitariam de auditoria, não importando, entretanto, a data certa da chegada do auditor em uma determinada dependência dentro do referíodo período de planejamento. Vale lembrar, que o planejamento dos serviços de auditoria do Banco é feito por órgão de cúpula e, em geral, as dependências não são informadas previamente da data em que serão auditadas.

O auditor, localizado inicialmente em sua praça-sede, é solicitado a efetuar uma rota, limitada por um tempo máximo e regressar à sua

praça-sede, quando, então, faria jus a um período de descanso remunerado. O processo se repetiria a cada período de roteamento, quando algumas das mais de quatro mil dependências do Banco seriam selecionadas para o trabalho de auditoria. O trabalho de rotear os auditores é, atualmente, feito de maneira empírica, sem o uso de qualquer teoria formal de Otimização e os auditores são roteados por funcionários com certa experiência no processo e com a ajuda de um mapa cartográfico, onde os auditores e agências são localizados visualmente. É interessante ressaltar que, na maioria das vezes, o roteamento é feito, dentro de um mesmo período de planejamento, de forma parcelada e, portanto, sem uma visão de todo o conjunto de agências e auditores envolvidos.

Estudamos o Problema dos Auditores com o objetivo de minimizar os custos envolvidos no processo de roteamento, tendo em vista a redução da distância total de roteamento e a utilização de auditores. Ainda que não tenhamos encontrado um problema idêntico na literatura, encontramos problemas semelhantes como o Problema do Caixeiro Viajante, Problema de Roteamento de Veículos (ou Problema da Distribuição), Problema de Coleta de Lixo Urbano, Problema de Escalonamento de Horários de Tripulação Aérea e outros mais, que podem ser vistos em Bodin et al [5].

Todos os problemas citados, anteriormente, são problemas de difícil resolução exata, para os quais a teoria de complexidade de algoritmos reservou uma classe especial, a classe do problemas NP-completos (Bornstein & Galdino [6]). E nesta classe se enquadra, também, o Problema de roteamento de Auditores que, em geral, se apresenta de grande porte, com mil agências e cerca de duzentas praças-sedes distintas. Diante de tais problemas, enquanto não se descobre algoritmos

exatos eficientes, utilizamos algoritmos heurísticos para obter soluções aproximadas para os mesmos. No capítulo II apresentamos alguns conceitos preliminares de complexidade computacional de algoritmos, classe de problemas **P** e **NP** e um método empírico para a avaliação do desempenho de heurísticas. Recorremos ao trabalho de Golden & Stewart [21] para desenvolver um método estatístico de comparação de duas heurísticas, utilizando os testes de hípoteses e significância da Estatística (Stevenson [33]), que podem ser vistos com maiores detalhes no ANEXO I. Tais testes são desejáveis, na medida em que, para todos os problemas que trataremos, os algoritmos produzem soluções aproximadas e, não se conhecendo, préviamente, as soluções exatas, teremos como único recurso compararmos as diversas soluções heurísticas.

No capítulo III apresentamos uma descrição de problemas da literatura que, pela sua similaridade com o Problema de Roteamento de Auditores, nos servirão de base para a construção de um modelo matemático e de um algoritmo para a solução deste problema. Começamos por apresentar o Problema do Caixeiro Viajante, segundo Hofman & Wolfe [24], o primeiro problema de Otimização Combinatória que teria surgido e, certamente, o mais conhecido entre todos. Um problema que, se por si só, não tem grande aplicabilidade, aparece como etapa intermediária fundamental na resolução de problemas com a referida aplicabilidade. Para o Problema do Caixeiro Viajante apresentamos dois algoritmos heurísticos, o algoritmo de Inserção (Bodin et al [5]) e o algoritmo de troca de arcos 2-otimal devido a Lin & Kernighan [26]. Programamos os dois algoritmos, em Turbo Pascal Borland 4.0, e apresentamos resultados computacionais, onde aplicamos o método estatístico do capítulo II para

avaliar o desempenho das heurísticas em teste.

O segundo problema apresentado no capítulo III é o Problema de Roteamento de Veículos 1-depósito. Tal problema pode ser encarado como uma extênsão do Problema do Caixeiro Viajante, onde um conjunto de veículos estacionado em um depósito central é solicitado a visitar um grupo de consumidores à espera de uma certa carga. Os veículos, aqui, são os análogos do Caixeiro Viajante e o objetivo do problema pode ser, entre outros, minimizar a distância total percorrida no roteamento (Christofides [13]). Para o Problema de Roteamento de Veículos 1-depósito apresentamos, também, dois algoritmos heurísticos, o clássico algoritmo de Clarke & Wright [8] e o algoritmo sequencial de Mole & Jameson [28]. Ambos utilizam o conceito de 'saving', com a diferença principal de que o primeiro gera rotas em paralelo e o segundo o faz sequenciamente. Apresentamos, ainda, uma versão inteligente para o algoritmo de Clarke & Wright devida a Golden et al [22], que utiliza o conceito de 'Heap' (Veloso e outros [36]) e 'Grid' que tornam o algoritmo original consideravelmente mais ágil. Programamos os dois algoritmos, em linguagem Turbo Pascal 4.0., apresentando uma série de resultados computacionais que comprovam, empíricamente, o desempenho de cada um deles.

Terminamos o capítulo III por apresentar o Problema de Roteamento de Veículos multi-depósito. Este pode ser encarado como sendo uma generalização do Problema de Roteamento de Veículos 1-depósito. Aqui, temos não mais um único depósito central, mas um conjunto deles, de onde partem e retornam os veículos. A dificuldade de resolução não é menor do que os dois problemas anteriores, ao contrário, a modelagem matemática deste problema é acrescida de restrições que o tornam mais complicado do que os os outros dois. Citamos os principais algoritmos

heurísticos eficientes (e são poucos) para o caso multi-depósito, são eles: o algoritmo de Gillett & Johnson [19], o de Wren & Holliday [37] e o algoritmo de Golden et al [22]. Este último é uma extênsão do algoritmo de Tillman & Cain [35] que, por sua vez, é uma extênsão do algoritmo de Clarke & Wright para problemas do tipo multi-depósito.

No capítulo IV descrevemos mais detalhadamente o Problema dos Auditores, comparando-o com os Problemas de roteamento apresentados no capítulo III e, em seguida, apresentamos a sua modelagem de forma semelhante ao Problema de Roteamento multi-depósito. Aqui, os auditores são os análogos dos veículos, as praças-sedes dos depósitos e não há carga alguma a transportar, mas sim, os análogos tempos de auditoria em cada dependência limitando o tamanho das rotas dos auditores. Ainda que exista semelhança entre este e aqueles problemas do capítulo III, uma série de restrições adicionais tornará o Problema de Auditores mais difícil do que aqueles.

Apresentamos, no capítulo IV, um algoritmo original para o Problema de Roteamento de Auditores, denominado SISROTAU. O algoritmo lança mão das teorias de Análise de Grupamento (Andeberg [2]) para agrupar os nós antes de promover o sequenciamento das rotas. Combinando uma variante do método de Forgy [16] (Realocação Iterativa, Lucas [27]), com uma, também, variante do algoritmo sequencial de Mole & Jameson [28], construímos o SISROTAU. Programamos o SISROTAU, em Turbo Pascal 4.0, e apresentamos uma bateria de testes com problemas encontrados na literatura, gerados aleatóriamente e problemas reais de roteamento de auditores do Banco do Brasil S.A. Aplicamos o nosso método estatístico, visto no capítulo II, para avaliar o desempenho do SISROTAU em relação a outros algoritmos. Para os problemas reais,

obtivemos os dados geográficos da rede de agências do Banco, junto ao IBGE-RJ, e comparamos os resultados obtidos pelo SISROTAU com aqueles obtidos pelo procedimento em uso no Banco.

No capítulo V, apresentamos as nossas principais conclusões sobre o trabalho, destacando a grande flexibilidade do SISROTAU que resolve tanto o problema 1-depósito, quanto o do tipo multi-depósito. Colocamos, também, algumas idéias para uma possível implementação do SISROTAU, sugerindo, inclusive, alguns procedimentos que podem ser acrescentados no sentido de tornar o sistema de roteamento mais eficiente e flexível a ponto de permitir uma maior iteração da máquina com o operador do sistema. Nas últimas páginas colocamos as nossas referências bibliográficas e o ANEXO II no qual listamos os dados relativos aos problemas reais de auditores que coletamos junto à PRESI-AUDIT, órgão da Direção Geral do Banco do Brasil S.A.

CAPÍTULO II

COMPLEXIDADE E AVALIAÇÃO DE DESEMPENHO DE ALGORITMOS HEURÍSTICOS

II.1- Problemas Intratáveis

II.1.1 - Introdução

Inicialmente, a Combinatória preocupava-se em sequenciar, agrupar, ordenar ou selecionar os objetos de um conjunto finito. A otimização acrescentou a preocupação com o "melhor" ou com o "ótimo". Os problemas que surgiram a partir de então, constituiram, por assim dizer, um desafio à capacidade humana de produzir novas teorias, algoritmos e tecnologia na produção de computadores cada vez mais velozes para resolvê-los em tempo hábil. Problemas como Caixeiro viajante, Roteamento de veículos, Recobrimento mínimo, Localização de armazéns em uma rede capacitada, Sequenciamento de processos industriais são alguns poucos exemplos da nova safra de problemas de difícil resolução que passaram a desafiar a inteligência de centenas de pesquisadores, motivados, muitas vezes, pela aplicabilidade de alguns dos referidos problemas a situações reais em que se desejaria racionalizar.

A convivência computacional com problemas de difícil resolução produziu o que se conhece hoje como a classe dos problemas **NP-COMPLETOS**. São os problemas intratáveis, ou seja, aqueles para os quais ainda não existem algoritmos de tempo polinomial. Em geral, um algoritmo é considerado bom quando sua complexidade de tempo é limitada polinomialmente em relação ao tamanho do conjunto de dados. A seguir, forneceremos alguns conceitos

preliminares sobre complexidade de algoritmos.

II.1.2- Complexidade de Algoritmos

Apresentaremos, inicialmente, o conceito de ordem de grandeza de uma variável ou função: dadas duas funções $f: \mathbb{N} \dashrightarrow \mathbb{R}_+$ e $g: \mathbb{N} \dashrightarrow \mathbb{R}_+$, a notação $f \in O(g)$, que se lê: " f é da ordem de g", significa que existe uma constante $k>0$ e $n_0 \in \mathbb{N}$, tal que $f(n) \leq k\, g(n)$, para todo $n > n_0$. Dizemos, também, que "g domina f assintoticamente". Assim é que, muitas vezes, apesar de não conhecermos uma expressão analítica para f, ou mesmo conhecer os seus valores, poderíamos apelar para o comportamento assintótico de f, desde que $f \in O(g)$ e $O(g)$ seja conhecida. Para exemplificar, consideremos as funções abaixo:

$f(n) = 300n^2 + 300n$ e $g(n) = n^3$,

$$f(n) = \left(\frac{300}{n^2} + \frac{300}{n} \right) n^3 \leq 600\, n^3 \text{ para } n > 1$$

Fazendo $k = 600$ e $n_0 = 1$ vem:

$n > n_0$ implica $f(n) \leq k \cdot g(n)$ e, portanto, dizemos que $f \in O(n^3)$.

A notaçao "O" é usualmente empregada para descrever a complexidade de algoritmos através de estimativas de tempo de execução e ocupação de memória. Para isto, imaginamos um computador fictício, onde todas as operações demandariam uma unidade de tempo, seja uma operação de comparação ou qualquer uma das operações aritméticas. Há um grande interesse em se relacionar a eficiência do algoritmo, ou seja, o tempo fictício que ele

tomaria quando submetido ao problema que mais esforço exigisse dele. Esta é a denominada "análise do pior caso". É interessante observar que esta teoria veio estabelecer um critério imparcial, no sentido de não depender da máquina, tampouco do problema, para se medir a eficiência de um algoritmo no que diz respeito à velocidade com que ele trabalha. Por outro lado, devemos ressaltar que a análise do pior caso não é, por si só, um instrumento definitivo para a análise de desempenho de algoritmos, posto que determinado algoritmo pode ter uma péssima análise de pior caso, entretanto, os problemas a que ele se destina não sejam semelhantes àqueles de pior caso.

Para ficar claro o conceito de complexidade computacional daremos dois exemplos simples, retirados de Bornstein & Galdino [6].

a)- Multiplicação de duas matrizes, quadradas, de ordem n:

$$C_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj} \quad , \quad i,j = 1,2,\ldots,n$$

poderíamos representar a multiplicação pelo algoritmo transcrito em linguagem Pascal:

```
for i:=1 to n do                          (I)
for j:=1 to n do
begin
   C[I,J]:= 0;
   for k:= 1 to n do
   C[I,J]:= C[I,J] + A[I,k] * B[k,J];    (II)
end.
```

Em (II) observamos n operações de soma e n

operações de multiplicação. Notamos, também, que tais operações sao necessárias à obtenção de cada elemento c_{ij} e, portanto, realizadas n^2 vezes. E o total de operações é:

$$n^2 \cdot n + n^2 \cdot n = 2n^3$$

A complexidade do algoritmo é $O(n^3)$. Estamos, evidentemente, supondo que as operações de multiplicação e soma demandam um mesmo tempo de máquina (fictícia) para serem realizadas e que tomamos como sendo igual a uma unidade.

b)-Busca Binária: dada uma lista ordenada crescentemente, representada pelo vetor A[1..n], de n elementos, deseja-se procurar o elemento "X" neste arranjo. Para tanto, vamos utilizar um procedimento de busca binária.

```
procedimento busca binaria;
begin
   menor:=1;
   maior:=n;
   encontrou:=false;
   while (menor<=maior) and (encontrou=false)  do
   begin
      T:= ⌊ (maior + menor)/2 ⌋;
      if A[T]=X then
      begin
         encontrou:=true;
         Apresente como saída o valor T
      end
      else
      if A[T]< X then menor:= T + 1
      else
      maior:= T - 1;
   end;
   if encontrou=false then
```

```
   Apresente como saída "ELEMENTO NAO ENCONTRADO"
end;{fim do procedimento}
```

A notação $\lfloor a \rfloor$ representa o maior inteiro q com a propriedade $q \leq a$. A complexidade deste algoritmo é definida pelo número de vezes em que o bloco "while" é executado, uma vez que em cada bloco "while" o algoritmo realiza um número constante de operações. É interessante observar que a este algoritmo se aplica perfeitamente o conceito de análise de pior caso, uma vez que a posição em que o elemento X se encontra na lista definirá o esforço da busca. Na primeira vez em que o bloco "while" é executado, a busca é feita sobre, aproximadamente, n/2 elementos e nas vezes consecutivas este total de elementos é sempre a metade do total da vez anterior. Concluimos, portanto, que o pior caso seria aquele em que o algoritmo chegasse ao final fazendo uma busca sobre uma lista com um único elemento, podendo este elemento ser o elemento "x" ou, também, o caso em que o elemento "x" não se encontra na lista. Assim temos:

$1^{\underline{a}}$ execução do bloco WHILE	$\simeq$	$n/2$
$2^{\underline{a}}$ execução do bloco WHILE	$\simeq$	$n/2^2$
|	|	|
k-ésima execução do bloco WHILE	$\simeq$	$n/2^k \simeq 1$

onde o símbolo "$\simeq$" representa "aproximadamente". Se na k-ésima execução temos $n/2^k \simeq 1$, então, $k \simeq \log_2 n$ e assim, a função de complexidade do algoritmo pode ser expressa por uma constante vezes $\log_2 n$, ou seja, $c.\log_2 n$. Finalmente, a complexidade do algoritmo de busca binária é $O(\log_2 n)$.

II.1.3 - As classes P e NP

Podemos definir um problema de Combinatória como um conjunto **I** de instâncias em que cada uma delas é um par **(V,C)**, onde **V** é o conjunto de soluções viáveis e **C: V** —>ℝ é uma função de custo. A manipulação de **V** e **C** pode ser feita através de dois algoritmos a_V e a_C . Diante de um candidato a solução do problema, "s", a_V testa se $s \in V$. Diante de uma solução viável, a_C retorna o valor C(s) que é o custo da presente solução viável. Agora, vamos introduzir as três formas com que pode ser tratado um problema combinatório. São elas as versões de decisão, identificação e otimização. A de decisão se enuncia assim:

"Dadas as representações dos parâmetros que definem **V** e **C** e um inteiro L, existe uma solução viável $s \in V$, tal que $C(s) \leq L$? "

Resolver um problema combinatório, colocado desta forma, consiste em decidir se é "SIM" ou "NÁO" a resposta à questão formulada. Na versão de identificação de um problema combinatório estamos interessados em identificar uma certa estrutura que satisfaça a um conjunto de propriedades:

"Dadas as representações dos parâmetros que definem V e C e um inteiro L, identifique uma solução viável S, tal que $C(s) \leq L$".

Finalmente, a versão de otimização tem a forma:

"Dadas as representações dos parâmetros que definem V e C, encontre a solução ótima".

Observamos que, em geral, resolver a versão de de otimização é mais difícil do que resolver a correspondente versão de decisão. Além do mais, quase sempre, a solução para a versão de otimização será, também, solução para as demais versões. Assim, obtida a solução ótima s^* de uma instância de um problema combinatório, teremos resolvido a correspondente versão de identificação se a

relação $C(s^*) \leq L$ é satisfeita , onde, então, s^* será a estrutura identificada. Além do mais, podemos responder "SIM" à versão de decisão.

O fato de a versão de decisão ser, em geral, mais fácil (ou não mais difícil) do que a correspondente versão de otimização, sugere que quaisquer resultados relativos a complexidade da versão de decisão de um dado problema, devem ser aplicados, pelo menos com a mesma ênfase, à versão de otimização. Em outras palavras, qualquer prova de sua possível intratabilidade pode ser estendida às outras versões. Daremos mais importância, portanto, à versão de decisão de um problema combinatório e a partir dela definiremos as classes **P** e **NP** de problemas combinatórios.

Definição: P é a classe dos problemas de decisão que podem ser resolvidos por algoritmos de tempo polinomial.

Assim, os dois algoritmos mostrados anteriormente, busca binária e multiplicação de matrizes, estão em P.

Ao invés de requerer um algoritmo de tempo polinomial para resolver todas as instâncias de um problema combinatório, colocado na sua versão de decisão, vamos exigir que o exame da validade de uma sugestão de solução para uma dada instância do problema, seja feito em tempo polinomial.

Definição: Se a representação de uma instância **SIM** de um determinado problema, colocado na versão de decisão, pode ser examinada por um algoritmo de tempo polinomial dizemos que o problema se encontra na classe **NP**.

Para ficar clara a definição de NP, ilustraremos com um exemplo:

Tomemos o problema do Caixeiro Viajante. Dados um Grafo completo G = (V,A) e um inteiro K > 0, existe em G um percurso de Caixeiro Viajante de peso ≤ K ? Dada uma instância SIM, (G = (V,A) , k), deste problema, o exame da validade de tal solução consiste em se construir um algoritmo que possa verificar se o percurso é hamiltoniano e se o seu peso é menor ou igual a k. Se tal algoritmo é polinomial, dizemos que o problema está em NP. De fato, o problema do Caixeiro Viajante está em NP.

Dentro da classe NP os pesquisadores têm dividido os problemas em duas classes principais, os problemas ditos **NP - COMPLETOS** E **NP-HARD.** Não querendo esgotar o assunto, diríamos que os dois últimos conceitos têm o mesmo significado prático, representam classes de problemas de difícil resolução, para os quais não se conhece algoritmo de tempo polinomial, ou seja, algoritmo considerado eficiente, para solucioná-los. Infelizmente, o problema que vamos abordar, Problema dos Auditores, encontra-se na classe **NP**, de forma semelhante ao problema de roteamento de veículos, cujos algoritmos exatos conseguem resolver apenas problemas pequenos de pouca aplicação prática.

II.2 - Avaliação de Desempenho de Heurísticas

Vimos anteriormente que alguns problemas, ditos intratáveis, não possuem algoritmos de tempo polinomial capazes de resolvê-los de maneira exata. Na prática duas coisas podem acontecer:

i)- O número de variáveis, que descrevem o problema que se pretende resolver, cresce de forma exponencial em relação aos dados de entrada do problema.

ii)-A complexidade computacional do algoritmo utilizado para a solução exata do problema é suficientemente elevada para inviabilizar o tempo de execução do algoritmo.

Em decorrência destes dois complicadores surgiram os algoritmos heurísticos. Os algoritmos heurísticos atuam no sentido de fornecer uma solução aproximada para problemas intratáveis de maneira exata. A história recente da Otimização Combinatória mostra um número cada vez maior de pesquisadores que se dedicam ao estudo de algoritmos heurísticos para os mais variados tipos de problemas de otimização [20]. Se, por um lado, os algoritmos heurísticos aparecem para solucionar problemas difíceis, apresentando, em geral, uma complexidade computacional menor do que os algoritmos exatos, eles possuem a desvantagem de não garantirem o ótimo. Face a esta situação há de se julgar os algoritmos heurísticos, entre outras coisas, pela qualidade das soluções produzidas, ou seja, gostaríamos de saber quão distantes do ótimo estarão as soluções geradas pelo algoritmo.

Destacamos dois principais métodos para analisar o desempenho de algoritmos heurísticos:

i)- Análise do pior caso
ii)- Análise empírica

A análise do pior caso tem por objetivo determinar o máximo desvio que determinada heurística poderia produzir em relação à solução ótima para determinado problema. Assim, de forma semelhante à análise do pior caso para a complexidade computacional, tal tipo de análise, apesar de ser sempre desejável, não pode refletir de maneira absoluta o comportamento de determinado algoritmo, posto que um determinado algoritmo

heurístico pode produzir, em média, boas soluções e ter uma péssima análise de pior caso. Mencionamos ainda, que a alaboração da análise de pior caso para um algoritmo heurístico é, quase sempre difícil e, conforme citam Hoffman & Wolfe [24], em muitos casos, tão difícil quanto resolver o prório problema combinatório.

Percebemos nos dias atuais um número cada vez maior de pesquisadores que se dedicam ao estudo da análise de pior caso para heurísticas, entre os quais destacamos, por exemplo, Marius M.Solomon [32] que trata de heurísticas para o Problema de Roteamento de Veículos.

Sem o rigor científico da Análise de pior caso, a Análise Empírica baseia-se fundamentalmente na experiência computacional. Além da preocupação evidente com a qualidade da solução, a Análise empirica relaciona os seguintes atributos de um algoritmo heurístico:

- tempo de processamento
- facilidade de implementação
- flexibilidade
- simplicidade

Os dois primeiros atributos dispensam qualquer explicação. Para entendermos melhor o atributo flexibilidade, vejamos um exemplo: consideremos dois algoritmos heurísticos, A e B, que resolvem o Problema do Caixeiro Viajante. O algoritmo A resolve tanto o caso simétrico quanto o assimétrico e o B resolve apenas o caso simétrico. Diríamos, portanto, que o algoritmo A é mais flexível do que o algoritmo B. O atributo simplicidade diz respeito aos recursos que determinado algoritmo faz uso, por exemplo, um algoritmo que utiliza tão somente as quatro operações aritméticas em toda a sua extensão é mais simples do que outro que, com a mesma

finalidade, lança mão de cálculos trigonométricos, logarítmicos e outros mais.

No que se refere a qualidade das soluções, dentro do contexto da Análise Empírica, Golden & Stewart [21] apresentam um método estatístico para a avaliação de desempenho de algoritmos heurísticos. O método é baseado no conceito de testes de hipóteses e significância da Estatística e é utilizado para comparar os resultados de duas ou mais heurísticas colocadas em teste. No ANEXO I nós apresentamos, em detalhes, conceitos básicos sobre os referidos testes de hipóteses, ilustrando a aplicação do método de Golden & Stewart [21] com alguns exemplos. Finalmente, ressaltamos que tal método nos será útil para a comparação ou avaliação de desempenho de diferentes algoritmos heurísticos que apresentaremos nos capítulos seguintes.

CAPÍTULO III

PROBLEMAS DE ROTEAMENTO

III.1 - Introdução

A literatura contém uma variedade enorme de problemas ligados a roteamento. Embora não tenhamos encontrado, especificamente, um problema como o problema dos Auditores, encontramos problemas semelhantes como Caixeiro Viajante, Roteamento de Veículos, Coleta de Lixo Urbano, Escalonamento de horário de tripulações etc... e todos são problemas combinatórios de difícil resolução. Dentre todos, sem dúvida, o mais estudado e, possivelmente, o primeiro que surgiu na literatura é o Caixeiro Viajante (Travelling Salesman Problem) [24]. O Problema do Caixeiro Viajante está intimamente ligado a outros problemas de grande aplicação prática, sendo na maioria das vezes, uma etapa intermediária da resolução destes mesmos problemas. Como exemplo, citamos o Problema de Roteamento de veículos, o Problema de Escalonamento de Horários de Tripulação Aérea e o Problema de Sequenciamento Ótimo de Processos Industriais. A seguir, nós estudaremos três problemas básicos, Caixeiro Viajante, Roteamento de Veículos 1-depósito e Roteamento de veículos multi-depósito, com o objetivo de fazer uma revisão da literatura existente, sob enfoque heurístico, efetuando, assim, a transição para o Problema dos Auditores.

III.2- O Problema do Caixeiro Viajante

Dado um conjunto de n cidades, {1,2,...,n}, consideremos o seguinte problema: um indivíduo parte da cidade 1, onde reside, e sai a visitar as demais cidades do conjunto, passando uma única vez por cada cidade e, então, retornando à cidade de partida. Encontre o percurso que minimize a distancia total percorrida por este indivíduo. Recorrendo à teoria de grafos, definiríamos o problema da seguinte maneira: Dado o Grafo G(V,A), não orientado, onde V é o conjunto de vértices existentes e A o conjunto de arcos, encontre o ciclo Hamiltoniano de menor peso neste grafo. Um ciclo Hamiltoniano é um caminho que começa e termina em um mesmo nó, passando por todos os demais nós uma única vez. O ciclo de menor peso para o problema em questão é denominado o ciclo Ótimo. O Grafo G(V,A) é dito não-orientado quando para qualquer par de nós i, j existem os arcos associados (i,j) e (j,i) de mesmo peso. Assim, a matriz de pesos ou distâncias associada ao grafo G(V,A) seria simétrica. A propósito, ao longo deste trabalho nós trataremos exclusivamente de problemas simétricos a menos que se diga o contrário. Um grafo G(V,A) é dito completo quando para todo par de nós i,j existe um arco que os une. Um grafo completo, não orientado, com n nós possui, portanto $\frac{n \cdot (n-1)}{2}$ arcos. Uma idéia bastante simples, porém de pouco efeito prático, seria a de contar todos os ciclos hamiltonianos existentes no grafo e, então, selecionar o de menor peso. A idéia se torna inviável diante da enormidade de ciclos hamiltonianos possíveis em um grafo. Por exemplo, em um grafo completo, não orientado, temos exatamente $\frac{(n-1)!}{2}$. Para um problema pequeno, de pouca aplicação prática, com 10 nós teríamos um

total de 181.440 ciclos hamiltonianos possíveis o que já é um número razoavelmente grande. O problema do Caixeiro Viajante pode ser formulado como um problema de programação inteira, senão vejamos:

$$\text{minimize} \quad \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} X_{ij} \qquad \text{(III.1)}$$

sujeito a

$$\sum_{i=1}^{n} X_{ij} = 1 \quad (j=1,..,n) \qquad \text{(III.2)}$$

$$\sum_{j=1}^{n} X_{ij} = 1 \quad (i=1,..,n) \qquad \text{(III.3)}$$

$$X = (X_{ij}) \in S \qquad \text{(III.4)}$$

$$X_{ij} = 0 \text{ ou } 1 \quad (i,j = 1,2,..n) \qquad \text{(III.5)}$$

$$d_{ii} = \infty \quad (i=1,2,..,n) \qquad \text{(III.6)}$$

Notação: n = o número total de nós da rede
V = conjunto de todos os nós

$$X_{ij} = \begin{cases} 1 \text{ se o arco (i,j) pertence à solução} \\ 0 \text{ se não pertence} \end{cases}$$

d_{ij} = distancia entre os nós i,j, medida sobre o arco (i,j)

A expressão (III.1) representa a função objetivo do problema de minimização. As restrições (III.2) garantem que em cada nó solução do problema incidirá um único arco. As restrições (III.3) garantem que de cada nó solução do problema sairá um único arco. As expressões (III.1) a (III.3) constituem o problema da designação (Assignment

Problem) mas não são suficientes para definir o problema do Caixeiro Viajante (PCV) uma vez que podem existir subciclos satisfazendo as mesmas mas não sendo, entretanto, um circuito hamiltoniano sobre a rede (fig III.1).

Para evitar a presença de subciclos acrescentamos as restrições (III.4). S é um conjunto selecionado especialmente para proibir as soluções que satisfazendo as restrições do problema da designação possuem subciclos. Na literatura aparecem várias maneiras de definir o conjunto S :

i)- $S = \{ (X_{ij}) : \sum_{i \in Q} \sum_{j \notin Q} X_{ij} \geq 1$
para todo subconjunto Q, não vazio e próprio de V.$\}$

ii)- $S = \{ (X_{ij}) : \sum_{i \in Q} \sum_{j \in Q} X_{ij} \leq |Q| - 1$

para todo subconjunto Q, não vazio, de $V - \{1\}$ $\}$

iii)-$S = \{ (X_{ij}) : y_i - y_j + nX_{ij} \leq n - 1$

para $2 \leq i \neq j \leq n$ e para qualquer número real y_i .$\}$

As formas (i) e (ii) apresentam aproximadamente 2^n restrições para evitar a formação de subciclos, enquanto a forma (iii) apresenta $n^2 - 3n + 2$ restrições com a mesma finalidade.

As restrições (III.5) definem a variável X_{ij} como sendo do tipo zero ou um e as restrições (III.6) evitam a formação de laços ou 'loops'.

Existem outras formas de modelar ou formular o problema do Caixeiro viajante seja como problema de programação inteira zero-um ou programas mistos, como pode ser visto em Padberg [29].

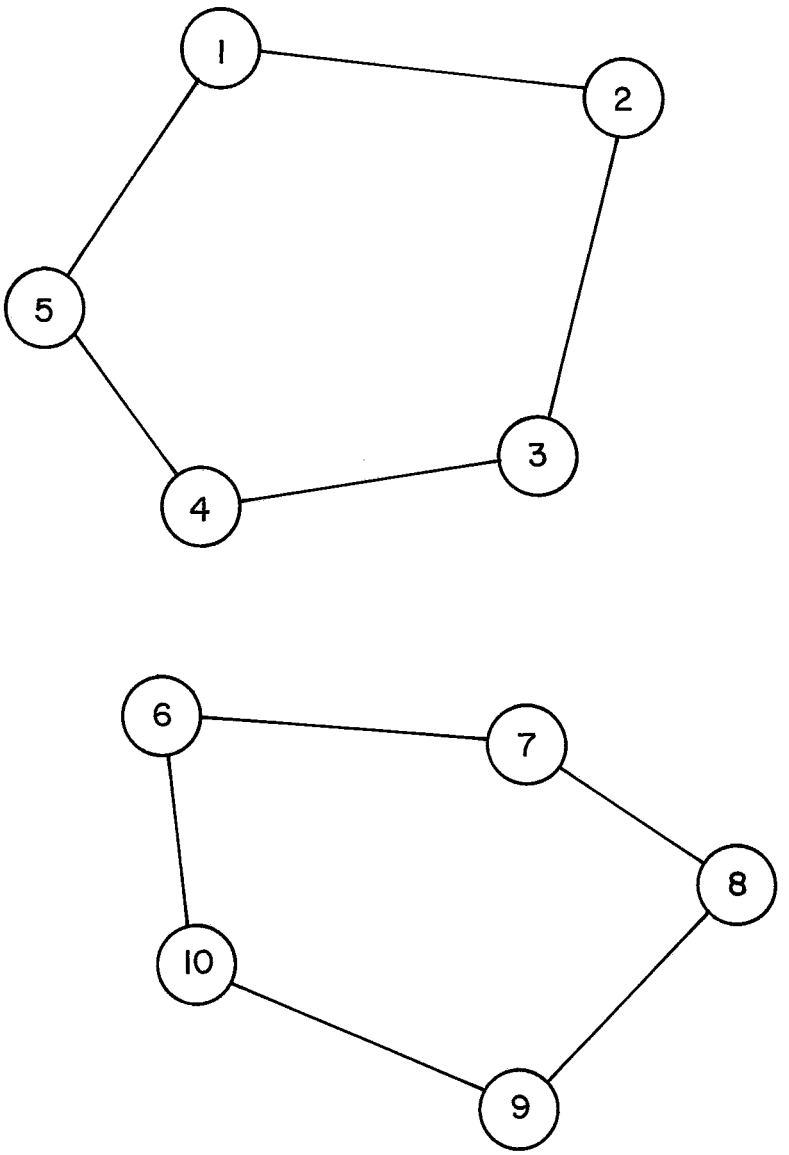


FIG. III.1 - SATISFAZ AS RESTRIÇÕES (III.2) E (III.3) MAS NÃO É CIRCUITO DE PCV.

Modelar o PCV é fácil, resolvê-lo, no entanto, tem-se constituído em um grande desafio para pesquisadores de todas as épocas. O problema está em NP, conforme mostram Lenstra et al [25], e até os dias atuais não se descobriu um algoritmo exato eficiente para solucioná-lo. A maioria dos algoritmos exatos existentes são do tipo 'Branch and Bound' ou métodos de enumeração implícita. Balas [4] cita que a tentativa de resolver o PCV, desde os trabalhos de Dantzig, Fulkerson e Johnson [1954,1959], têm contribuído para a formação de novas teorias para a Otimização Combinatória e cita, como exemplo, o método de enumeração (Branch and Bound e Enumeração Implícita) como sendo uma destas contribuições. No mesmo artigo, Balas apresenta os principais algoritmos exatos para o PCV, a maioria deles baseado em relaxações dos problemas de Designação (Assignment Problem) e Árvore Geradora de Peso Mínimo (Minimal Spanning Tree) que fornecem limites inferiores para o PCV. Para darmos um exemplo, vejamos um limite inferior que se obtem do problema da árvore geradora de peso mínimo,dado por Cristofides [11]: Dada uma matriz C de custos (ou distancias) associada a um grafo não direcionado, completo, com n vértices, vamos supor que o arco (X_1,X_2) está presente na solução ótima do problema do Caixeiro Viajante sobre o grafo. Se retirarmos o arco (X_1,X_2) da solução do PCV, obteremos um caminho hamiltoniano e, portanto, sem ciclos, com n-1 arcos. Evidentemente, tal caminho é uma árvore geradora para o grafo e o seu peso ou custo deve ser, necessáriamente, maior ou igual ao peso da árvore geradora de peso mínimo associado ao mesmo grafo. Denotando o valor da solução ótima do PCV por O(PCV) e o ótimo do problema de árvore geradora mínima por O(AGM) temos, evidentemente, a seguinte relação:

$$O(AGM) \leq O(PCV) \qquad \text{(III.7)}$$

Cristofides coloca, ainda, um limite inferior melhor, derivado do problema de árvore geradora mínima:

$$O(AGM) + \max_{X_i} [\ C(X_i,S)\ ] \leq O(PCV) \qquad (III.8)$$

onde S é o segundo vértice mais próximo ao vértice X_i .

III.2.1 - Alguns Algoritmos Heurísticos para o PCV

Um número muito grande de heurísticas para o PCV tem aparecido na literatura e uma relação das principais pode ser vista em Bodin et al [5]. Apresentamos a seguir dois algoritmos heurísticos para o PCV e os resultados computacionais associados.

II.2.1.a - Algoritmo de Inserção

O algoritmo da inserção é dito ser construtivo, uma vez que ele parte de um nó inicial, qualquer, e vai aumentando gradativamente o subciclo formado até que todos os nós estejam sobre um mesmo ciclo hamiltoniano, ou seja, um circuito de PCV. Em cada passo do algoritmo, como veremos mais adiante, dois procedimentos são considerados: i)- a escolha do nó que será incorporado ao subciclo emergente e ii)- a melhor posição de inserção deste nó no subciclo. O primeiro procedimento permite algumas variações do método de inserção e Bodin [5] relaciona algumas delas:inserção do vizinho mais próximo (IMP),

inserção mais barata (IMB), inserção arbitrária (IMA) e inserção do vizinho mais distante (IMD). Apresentaremos aqui, os métodos IMP e IMD:

Algoritmo IMP:

passo 1: comece a rota com um subgrafo consistindo de um único nó i.

passo 2: encontre um nó k tal que C_{ik} é mínimo e forme o subciclo (i - k - i).

passo 3: passo de seleção: encontre o nó K, ainda não roteado, o mais próximo possível do corrente subciclo.

passo 4: passo de inserção: encontre o arco (i,j) no subciclo, que minimiza $C_{ik} + C_{kj} - C_{ij}$. Insira o nó K entre i e j.

passo 5: Se já está formado um ciclo hamiltoniano, pare. Senão, volte ao passo 3.

C é a matriz de custos ou pesos associada ao grafo em que se deseja obter o circuito de PCV. A distância referida no passo 3, ou seja, distância de um nó não roteado ao subciclo pode ser assim representada:

$D(j) = \min_{k} \{C(j,k)\}$, onde $j \in (V-V_T)$ e $k \in V_T$.

onde

V = conjunto de vértices do grafo

V_T= conjunto dos vértices já incluidos no subciclo

ou em palavras, a distância de um nó j, não pertencente ao subciclo, ao subciclo é dada pela

distância de j ao seu nó mais próximo e pertencente ao mesmo subciclo.

O método de inserção do vizinho mais distante, IMD, difere do IMP, tão somente, no critério de inserção. Assim, para o IMD teríamos as seguintes modificações:

passo 2: encontre um nó k tal que C_{ik} é máximo e forme o subciclo (i-k-i).

passo 3: passo de seleção: encontre o nó K, ainda não roteado, o mais distante possível do corrente subciclo.

A complexidade computacional de ambas as versões, IMP e IMD, é da ordem de n^2 e Bodin [5] apresenta a análise de pior caso para as duas versões:

IMP: $\dfrac{\text{solução heurística IMP}}{\text{solução ótima}} \leq 2$

IMD: $\dfrac{\text{solução heurística IMD}}{\text{solução ótima}} \leq \log(n) + 1$

A análise acima é válida para PCV que satisfaz a desigualdade triangular:

$$C_{ij} \leq C_{ik} + C_{kj}$$

para qualquer i,j,k $\in$ V. Como exemplo, podemos citar os problemas que trabalham com distancias euclideanas ou matriz de caminhos mínimos [30]. Ademais, neste trabalho, todos os problemas apresentados satisfazem a citada desigualdade triangular.

III.2.1.b- Método de troca de arcos

Ao contrário do método da inserção, o método de troca de arcos exige uma solução inicial qualquer para dar partida no algoritmo. Tal método é por vezes denominado 'heurística de otimização local' ou 'heurística de melhoramento de soluções'. Bodin [5] cita que as heurísticas de troca de arcos são as mais conhecidas para o PCV e quem primeiro introduziu a idéia foi Lin (1965) com as rotinas 2-otm e 3-otm. Posteriormente, Lin e Kernighan [26] publicaram uma rotina de troca de arcos k-otm. Uma rotina de troca de arcos tem o seguinte funcionamento básico:

passo 1: gere um circuito de PCV inicial

passo 2: melhore o circuito utilizando uma das rotinas de troca de arcos (2-otm, 3-otm, ...,k-otm)

passo 3: continue no passo 2 até que a solução não possa mais ser melhorada

Uma rotina do tipo k-otm efetua todas as trocas viáveis de k arcos que reduzem o valor da função de custo associada ao problema do Caixeiro Viajante. A solução final produzida pelo algoritmo é dita uma solução k-otimo, no sentido de que nenhuma troca posterior, de k arcos entre si, poderá melhorar a solução que é um ótimo local. A qualidade das soluções produzidas por este método depende da solução inicial fornecida e isso poderemos sentir quando apresentarmos os resultados computacionais para o PCV. Uma rotina (k+1)-otm produz resultados melhores ou iguais a rotinas do tipo k-otm mas a complexidade deste algoritmo cresce

exponencialmente com o número de arcos que se deseja trocar. A figura III.2 nos dá uma idéia geométrica do funcionamento de um algoritmo 2-otm. Na parte (a) da figura III.2 os arcos AB e ED serão retirados da solução e trocados pelos AD e BE como mostra a parte (b) da mesma figura. O algoritmo de inserção, apresentado anteriormente, e o de troca de arcos são definidos tanto para o PCV simétrico quanto para o assimétrico, embora, neste trabalho, lidamos tão somente com problemas simétricos.

Algoritmo 2-ótimo

Consideremos um ciclo de PCV dado pelo conjunto $H = \{ X_1, X_2, \ldots, X_n \}$ de seus arcos não direcionados. Seja $X = \{ X_i, X_j \}$ o conjunto formado por dois arcos pertencentes a H e que pretendemos trocar com os arcos $Y = \{Y_p, Y_q\}$, desde que haja uma melhoria da função objetivo, ou seja, uma redução no custo total do circuito. Uma vez efetuada a troca, teríamos $H_1 = (H - X) \cup (Y)$ representando o novo circuito, melhorado pela troca dos arcos. É fácil verificar que em circuito hamiltoniano com n arcos existem $(C_n^2 - n)$ trocas, dois a dois, de arcos possíveis. O valor (n) que se desconta é devido aos arcos adjascentes que, quando 'deletados' da corrente solução, não podem dar origem a outros arcos distintos e, então, não são considerados para troca. Vamos denotar por δ o valor economizado em uma destas trocas:

$$\delta = C(H) - C(H_1) = C(X_i) + C(X_j) - C(Y_p) - C(Y_q)$$

onde C representa o custo total da solução ou o custo de determinado arco.

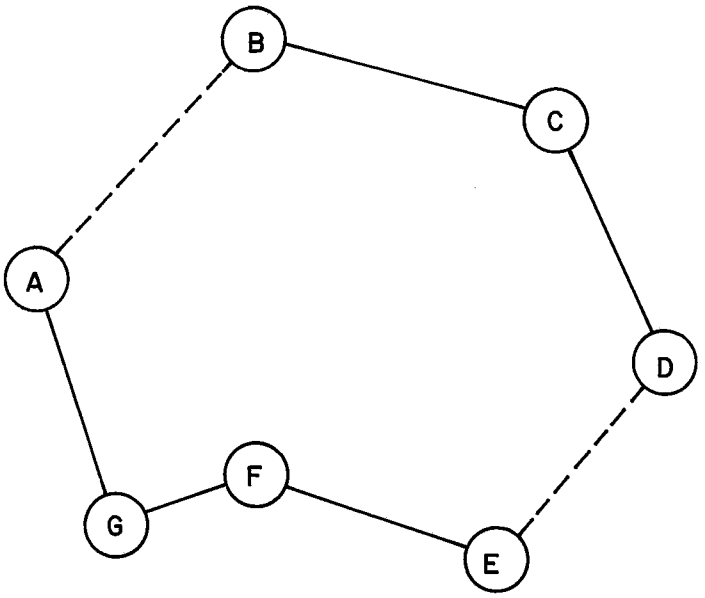


FIG. III.2.a

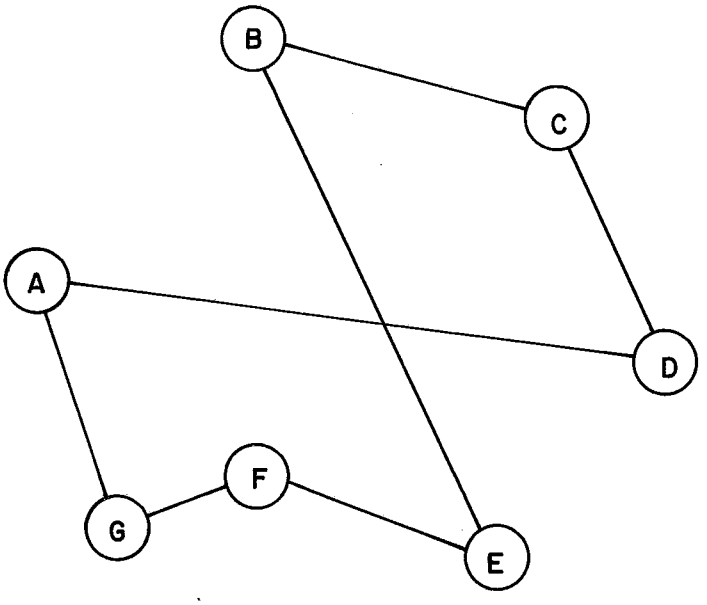


FIG. III.2.b

O algoritmo 2-otimo examina, a cada iteração, a melhor troca possível, guardando a nova solução decorrente da troca e o valor de δ_{max} associado a esta melhor troca. Se $\delta_{max} = 0$ o algoritmo termina. Caso contrário, repete-se todo o procedimento até que a solução não possa mais ser melhorada. A seguir apresentamos, em pseudo-linguagem Pascal, o algoritmo 2-otimo:

```
Begin
   tome H = { X1,X2,...,Xn } como solução inicial.
   Repeat
      δmax := 0
      For i:= 1 to (n-2) do
      for j:= (i+2) to n or (n-1) do
      if [C(Xi)+C(Xj)-(C(Yp)+C(Yq))] > δmax then
      Begin
         δmax := (C(Xi) + C(Xj) - C(Yp) - C(Yq))
         Guardar i e j
      end;
      if δmax > 0 then H:=H - (Xi,Xj) ∪ (Yp,Yq)
   Until δmax = 0
end.
```

Na sexta linha do algoritmo a opção j = n-1 é permitida somente para i = 1. Outro ponto importante que deve ser notado é que os arcos Y_p e Y_q ficam perfeitamente determinados, e são únicos, quando da deleção dos arcos X_i e X_j. Na verdade, quando deletamos os arcos X_i e X_j, não adjacentes, o circuito fica reduzido a dois caminhos não conectados entre si, dando origem a quatro extremidades que podem ser ligadas de duas maneiras diferentes, mas uma delas não forma um circuito de PCV, restando, portanto, uma alternativa viável, que forma um circuito de PCV. Em geral as rotinas de troca de arcos, k-otm, são utilizadas em composição com outros algoritmos que fornecem boas soluções iniciais para a rotina k-otm. Assim, podemos combinar IMP + 2-otm, IMD + 2-otm e

verificar se a rotina 2-otm consegue melhorar uma solução inicial fornecida. Nos problemas de roteamento de veículos, que apresentaremos mais adiante, também utilizamos estas rotinas de troca de arcos para tentar melhorar, localmente, alguns circuitos de PCV, sub-problema do problema de roteamento de veículos.

III.2.2 - Resultados Computacionais para o PCV

Programamos os algoritmos IMP,IMD e 2-otm conforme Syslo [34], modificando um pouco a estrutura de dados no sentido de economizar memória. A tabela III.4 apresenta os resultados computacionais obtidos para trinta PCV's, com 100 nós cada um, gerados aleatóriamente a partir de suas coordenadas retangulares, X e Y. Tais coordenadas são tomadas, com valores inteiros, nos intervalos [10,310] e [10,190], respectivamente, sob uma distribuição uniforme de probabilidades de se tomar qualquer um dos valores, nos respectivos intervalos. Na coluna AGM utilizamos a expressão (III.8) para obter um limite inferior para o PCV, a partir da árvore geradora mínima associado ao respectivo grafo. Nas colunas IMD, IMD/2otm, IMP e IMP/2otm apresentamos os desvios percentuais, em relação à coluna AGM, para os outros métodos. Vale a pena lembrar que IMD/2otm e IMP/2otm representam, respectivamente, os métodos IMD e IMP em composição com a rotina 2-otm de troca de arcos. Os valores colocados entre parênteses representam os tempos, em segundos, de processamento em um microcomputador PC-XT. Na última linha da mesma tabela colocamos os desvios médios para os métodos, bem como os tempos médios de processamento. A tabela evidencia a superioridade do método IMD sobre o IMP e do IMD/2otm sobre o IMP/2otm. Nota-se, também, que a

NUM	AGM	IMD	IMD/2otm	IMP	IMP/2otm
1	1485	23.5	22.7	41.3	29.6
		(8)	(20)	(8)	(77)
2	1653	20.1	20.0	31.0	25.6
		(8)	(10)	(9)	(41)
3	1583	19.0	15.2	32.8	25.3
		(8)	(20)	(8)	(46)
4	1587	15.7	14.6	34.2	30.2
		(8)	(15)	(8)	(46)
5	1566	17.2	15.1	30.9	16.1
		(8)	(25)	(8)	(154)
6	1630	21.0	19.8	33.0	24.2
		(8)	(15)	(8)	(87)
7	1570	20.1	20.1	30.1	25.1
		(8)	(5)	(8)	(66)
8	1573	26.6	26.3	42.4	26.6
		(8)	(10)	(8)	(113)
9	1519	20.3	19.4	35.7	28.5
		(8)	(20)	(8)	(56)
10	1640	18.9	18.4	35.0	21.3
		(8)	(10)	(8)	(107)
11	1540	18.0	17.5	42.0	21.6
		(8)	(20)	(8)	(179)
12	1597	17.1	17.1	35.6	24.6
		(8)	(5)	(8)	(71)
13	1598	17.7	17.7	34.9	21.0
		(8)	(10)	(8)	(102)
14	1697	17.8	17.8	39.0	23.6
		(8)	(10)	(8)	(107)
15	1578	14.2	13.5	34.6	28.8
		(8)	(15)	(8)	(46)
16	1530	25.4	24.9	41.7	33.0
		(8)	(15)	(8)	(97)
17	1503	22.2	20.7	43.5	32.2
		(8)	(25)	(8)	(92)
18	1578	19.6	19.1	36.9	27.6
		(8)	(15)	(8)	(72)
19	1567	21.6	20.4	31.2	23.7
		(8)	(15)	(8)	(51)
20	1543	17.4	17.3	39.0	21.0
		(8)	(10)	(8)	(159)
21	1580	28.9	27.7	37.4	24.8
		(8)	(15)	(8)	(61)
22	1476	11.7	11.3	36.5	19.9
		(8)	(10)	(8)	(128)

TABELA III.4 - Problemas de Caixeiro Viajante

23	1476	15.8	15.6	45.1	21.9
		(8)	(10)	(8)	(128)
24	1603	20.9	18.4	33.0	21.0
		(8)	(35)	(8)	(97)
25	1621	19.7	17.9	31.7	28.6
		(8)	(25)	(8)	(30)
26	1578	25.1	23.8	39.7	31.8
		(8)	(20)	(8)	(61)
27	1478	24.0	23.5	41.8	28.3
		(8)	(10)	(8)	(118)
28	1568	19.7	14.9	33.8	23.4
		(8)	(51)	(60)	(77)
29	1602	21.4	17.2	34.1	23.0
		(8)	(36)	(8)	(77)
30	1631	20.7	20.1	34.6	29.1
		(8)	(30)	(8)	(51)
Media	-	20.0	19.3	36.4	25.4
		(8)	(17.7)	(8)	(86.5)

Tabela III.4 - Continuação

qualidade do refinamento produzido pela rotina 2-otm depende fortemente da solução inicial que lhe é fornecida, o que pode ser visto comparando-se as colunas IMD/2otm e IMP/2otm.

Vamos, agora, aplicar o teste de hipóteses para médias, citado no capítulo II e detalhado no ANEXO I, para as colunas IMD/2otm e IMP/2otm. Inicialmente tomamos as diferenças d_i entre as colunas IMD/2otm e IMP/2otm, nesta ordem, e fazemos:

H_0: $\bar{d} = 0$ (hipótese nula: os algoritmos IMD/2otm e IMP/2otm produzem resultados iguais para toda a população de PCV's)
H_1: $\bar{d} < 0$ (hipótese alternativa: o algoritmo IMD/2otm produz soluções melhores do que IMP/2otm)

$\alpha = 5\ \%$ (nível de significância do teste)

O desvio médio da amostra é dado por:

$\bar{d}_x = -6.71$

Calculemos a estatística t de Student:

$$t = \frac{\bar{d}_x}{s_d / \sqrt{n}}$$

$$s_d = 4.404$$

$$t = - 8.204$$

Recorrendo à tabela para a distribuição de Student e com os parâmetros $\nu = 29$ (grau de liberdade) e $\alpha = 5\ \%$, encontramos o valor crítico:

$$t_c = - 1.699$$

Temos evidentemente $t < t_c$ e, portanto, o valor da da estatística t caiu na região de rejeição e, então rejeitamos a hipótese de que os algoritmos são iguais e mais, concluimos que o IMD/2otm produz melhores resultados do que o IMP/2otm.

É importante ressaltar que os resultados mostrados sob a coluna AGM foram obtidos a partir de uma modificação no algoritmo de PRIM, apresentado em Syslo [34], originalmente concebido para resolver o problema de Árvore Geradora Mínima em um grafo. Ao programa que reune as codificações dos algoritmos IMP, IMD e 2-otm daremos o nome de TSP, enquanto aquele que obtém o limite a partir da AGM designaremos por LIMAGM.

III.3 - Problemas de Roteamento de Veículos

A designação "Problemas de Roteamento de Veículos" é a tradução das designações "Vehicle Routing", "Vehicle Scheduling", "Truck Dispatching" e "Delivery Problem" encontradas na literatura especializada de língua inglesa. Alguns destes problemas tratam de distribuir ou recolher produtos(derivados do petróleo, correspondência postal, produtos alimentares, lixo doméstico e industrial, etc.), outros tratam do transporte de pessoas(transportes especiais para deficientes físicos, transportes escolares,etc.) e outros, ainda, tratam de situações em que não há bens ou pessoas a transportar, mas, sim, operações a efetuar(inspeção de redes ou condutos, seleção de produtos num armazém,etc.). Colocaremos nesta categoria, também, o Problema de Roteamento de Auditores do Banco do Brasil S.A. A enorme variedade de situações englobadas conduz, naturalmente, a uma multiplicidade de modelos.

Assim, consideremos um conjunto de n localizações geográficas, N = { 1,2,...,n }, a primeira representando um depósito central onde se encontra estacionada uma frota de veículos e as demais representando os clientes. Para cada cliente, além de conhecida sua localização, é conhecida a sua demanda q_i, i = 2,...,n, relativa ao bem a ser distribuído. Para cada veículo, k, é conhecida a sua capacidade Q_k, k = {1,2,...,M}. Trata-se, portanto, de determinar as configurações das rotas a serem efetuadas pelos veículos, de modo a que cada cliente seja servido por um e somente um veículo, minimizando-se o comprimento (ou custo) do percurso total. Esta situação pode comportar diversas restrições adicionais, tais como:

a)-Restrições Temporais: Os veículos só podem

operar durante intervalos de tempo de duração limitada. Assim, cada rota não pode ter uma duração de tempo superior a um valor pré-fixado. Na avaliação de duração da rota, dependendo da situação, podem ser incluídos apenas os tempos de trajeto ou ser também incluídos os tempos de descarga e entrega das mercadorias.

Uma segunda situação é aquela em que os clientes só aceitam as encomendas durante certos intervalos de tempo (na literatura inglesa é designado por 'time-windows'). Neste caso, pode ainda ser considerada a situação em que o veículo chegue antes do tempo admissível e fique estacionado, esperando o início do serviço de descarga da mercadoria. Uma completa revisão Bibliográfica destas situações é mostrada em Golden & Assad [20].

b)-Restrições de Precedência: Em alguns casos impõe-se restrições de precedência entre clientes, não podendo determinados clientes serem visitados sem que outros tenham sido. Tal situação é comum em problemas que tenham distribuição e recolha simultânea de mercadorias.

c)-Restrições na Frota: O número de veículos que compõem a frota pode ser conhecido a priori, tendo nesse caso que impôr-se a restrição adicional de que o número de rotas a ser gerado não pode ultrapassar o número de veículos disponível.

O Problema de Roteamento de Veículos pode também ser generalizado, sendo as generalizações mais frequentes as seguintes:

i)-Múltiplos Depósitos

Há duas situações a considerar: a primeira é aquela em que existem vários depósitos onde a

frota fica estacionada, mas cada veículo está designado para um depósito específico e, portanto, a sua rota começa e termina em um mesmo depósito. A segunda situação é aquela em que um determinado veículo pode iniciar a sua rota em um depósito e terminá-la em outro. Pode-se impor, neste segundo caso, restrição relativa à capacidade máxima de estacionamento nos depósitos.

ii)-Frota Não-Homogênea

Muitas vezes temos um problema em que a frota de veículos não é homogênea, seja porque os veículos têm capacidades distintas ou até mesmo porque eles são diferenciados por compartimentos especiais de armazenagem.

iii)-Múltiplos Objetivos

Além do objetivo de minimizar o comprimento total, podem estabelecer-se outros objetivos. Assim, quando o número de veículos na frota não é definido a priori, posto que se admite a compra ou o aluguel de novos veículos, podemos colocar na função objetivo uma expressão que traduza este fato. Teremos, portanto, um problema de minimizar o custo total, representado pela soma dos custos variáveis ou de transporte e o custo fixo de aquisição ou aluguel de novos veículos. Ademais, aparecem problemas em que não se exige que todos os clientes sejam visitados, mas sim uma parcela de maior relevância.

Uma revisão bibliográfica bastante extensa sobre o Problema de Roteamento de Veículos e suas diferentes versões pode ser vista em [5]. A seguir, apresentaremos o Problema de Roteamento de Veículos, caso 1-depósito e multi-depósito, apresentando alguns algorítimos heurísticos para a sua resolução. Definiremos em cada situação um

problema básico em que aparecem apenas uma porção das diversas restrições citadas anteriormente.

III.3.1 Problema Básico de Roteamento de Veículos - único depósito central

Consideremos um conjunto de n localizações N = { 1,2,...,n }, a primeira representando um depósito central, onde está localizada uma frota homogênea de veículos k = 1,2,...,M com a mesma capacidade Q. Os demais nós, 2,3,...,n, representam os diversos clientes de capacidade q_i, i=2,3,...,n, que serão servidos pela frota e C = (C_{ij}) uma função de distância (ou custo) associada aos pares de nós i e j. Exigimos que cada cliente seja servido por um único veículo e temos por objetivo minimizar a distância total percorrida pela frota. Doravante, designaremos tal problema por Problema Básico de Roteamento de Veículos - caso 1-depósito, PRV1. Apresentamos, agora, uma formulação em programação matemática devida a Golden et all [22]:

$$\text{Min} \quad \sum_{k=1}^{M} \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} X_{ij}^{k} \qquad \text{(III.9)}$$

s.a.

$$\sum_{i=1}^{n} \sum_{k=1}^{M} X_{ij}^{k} = 1 \qquad j=2,\ldots,n \qquad \text{(III.10)}$$

$$\sum_{j=1}^{n} \sum_{k=1}^{M} X_{ij}^{k} = 1 \qquad i=2,\ldots,n \qquad \text{(III.11)}$$

$$\sum_{i=1}^{n} \left(X_{ij}^{k} - X_{ji}^{k} \right) = 0 \qquad k=1,..,M;j=1,...,n \qquad \text{(III.12)}$$

$$\sum_{j=1}^{n} X_{1j}^{k} \leq 1 \qquad k=1,...,M \qquad \text{(III.13)}$$

$$\sum_{i=2}^{n} q_i \left(\sum_{j=1}^{n} X_{ij}^{k} \right) \leq Q \qquad k=1,...,M \qquad \text{(III.14)}$$

$$X_{ij}^{k} \in \{0,1\} \qquad i,j=1,...,n;k=1,...,M \qquad \text{(III.15)}$$

$$X \in S \qquad \text{(III.16)}$$

onde

1 - nó que representa o depósito central
n - número total de nós consumidores
M - número total de veículos
q_i - quantidade de carga demandada pelo consumidor i
Q - capacidade máxima de carga por veículo

$$X_{ij}^{k} = \begin{cases} 1, \text{ se o arco (i,j) está na rota do k-ésimo veículo} \\ 0, \text{ caso contrário} \end{cases}$$

S- conjunto de todos os subciclos que contém o depósito '1'.

A função objetivo (III.9) traduz a minimização da distância total percorrida pela frota. As restrições (III.10) a (III.12) garantem que cada cada cliente é visitado por um único veículo e que este sempre saia de um nó no qual tenha entrado. As restrições (III.13) asseguram que cada veículo não

efetuará mais do que uma rota. As restrições (III.14) limitam a carga do veículo ao máximo Q. As restrições (III.15) definem as variáveis como sendo do tipo 0 ou 1. As restrições (III.16) evitam a formação de subcircuitos que não contenham o depósito central 1 e, de forma semelhante ao PCV, podem ser expressas de diversas formas.

III.3.1.1 - Algoritmos Heurísticos para o PRV1

Muitos são os algoritmos heurísticos que aparecem na literatura para o PRV1. Alguns puramente heurísticos, cujo critério de solução é, em geral, motivado por considerações geométricas, outros, resultantes de processo de otimização incompleta em algoritmos do tipo Branch and Bound, onde não se examinam todas as soluções que seriam examinadas no processo exato. Do primeiro tipo citamos, como exemplo, os algoritmos de Clarke & Wright [8], Mole & Jameson [28], Wren & Holliday [37] e Gillett & Miller [18]. Do segundo tipo citamos o trabalho de Christofides et all [12]. Ainda sobre os algoritmos do primeiro tipo, citados anteriormente, existem algumas diferenças básicas quanto a forma de construção das rotas. Alguns constroem as diversas rotas simultâneamente, ou em paralelo, outros procedem a construção sequencial das rotas, ou seja, uma nova rota será começada tão somente quando a rota anterior estiver terminada. O algoritmo de Clarke & Whrigt pode ser colocado em ambas as versões, paralelo e sequencial. Em contrapartida, o algoritmo de Mole & Jameson apresenta-se na versão sequencial.

Bodin et al [5] estabelecem uma classificação para os diversos algoritmos que resolvem o PRV1 de acordo com a estratégia que utilizam. A maior parte destes algoritmos pode ser enquadrada em uma ou mais das seguintes classes:

a)-Agrupar primeiro/sequenciar depois
b)-Sequenciar primeiro/agrupar depois
c)-Saving/inserção
d)-Métodos de troca de arcos ou melhoramento
e)-Métodos derivados da Programação Matemática
f)-Métodos iterativos homem/máquina
g)-Métodos exatos

Apresentaremos a seguir dois algoritmos para o PRV1 que se enquadram no tipo Saving/inserção. O primeiro é devido a Clarke & wright [8] e modificado por Golden et al [22]. O segundo emprega uma generalização do conceito de saving e é devido a Mole & Jameson [28].

III.3.1.2 Algoritmo tipo Saving para o PRV1

Em 1964 G.Clarke e J.W.Wright apresentaram um algoritmo heurístico para o PRV1, baseado no conceito de poupança (Saving em inglês), que pela sua eficiência e, principalmente, pela sua simplicidade vem sendo utilizado até os dias atuais. O algoritmo parte de uma solução inicial, trivial, onde é designado um veículo para atender cada cliente existente. A solução é, portanto, inviável, uma vez que se exige uma frota de tamanho igual ao número de clientes. A figura III.5 representa tal situação, com os clientes i e j sendo atendidos por dois veículos a partir do depósito 1. Se desejássemos utilizar um veículo somente, nós poderíamos eliminar um dos veículos e colocar os dois nós, i e j, sobre uma mesma rota, como pode ser visto na figura III.6. Assim, teríamos obtido uma poupança ou 'saving', representado por:

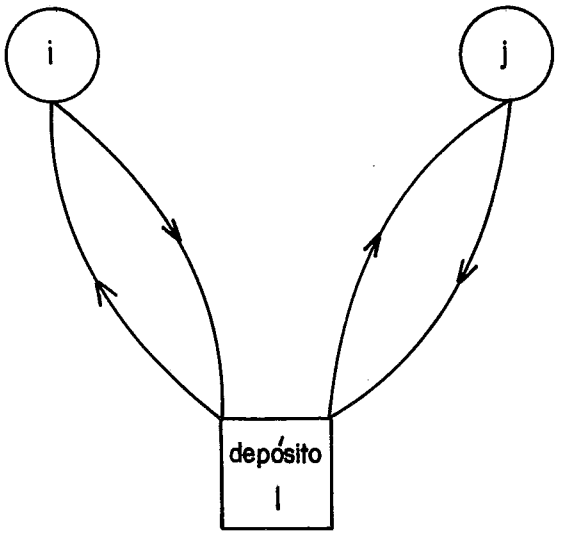


FIG. III.5 - DUAS ROTAS DISTINTAS SERVINDO OS NÓS i E j.

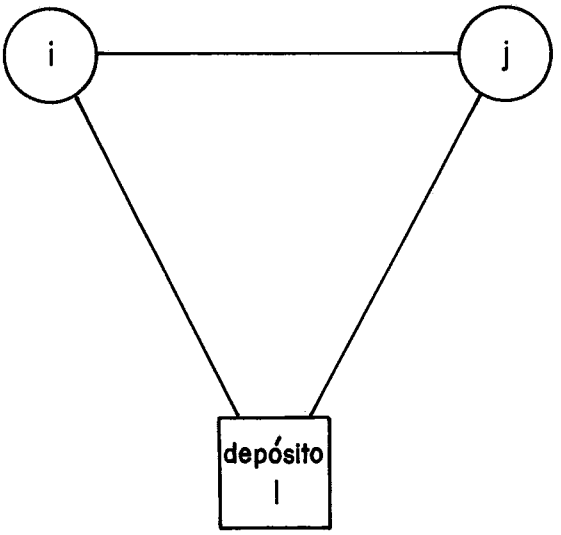


FIG. III.6 - UMA ÚNICA ROTA SERVE OS NÓS i E j.

saving = $(2d_{1i} + 2d_{1j}) - (d_{1i} + d_{1j} + d_{ij})$ ou
saving = $S_{ij} = d_{1i} + d_{1j} - d_{ij}$

Para todo par de nós, i e j distintos, existe um saving S_{ij} correspondente, dado pela expressão acima. O algoritmo original de Clarke & Wright calcula todos os possíveis savings, colocando-os em uma lista ordenada com o maior elemento no topo, para
uma rota já inicializada. Um ponto é dito ser interior quando ele não está ligado ao depósito;

iii)- o número máximo de veículos disponíveis não é excedido;

iv)- a capacidade do veículo não é excedida;

A construção das rotas se dá de forma paralela, ou seja, ao percorrer a lista, duas ligações sucessivas podem ser feitas em rotas distintas. Uma observação se faz interessante sobre a restrição ii. Tal restrição permite que se ligue, a cada passo do algoritmo, somente pontos extremos das rotas, ou seja, aqueles pontos que estão ligados diretamente ao depósito. Este é um ponto fundamental que difere este algoritmo do de Mole & Jameson, aonde se permitirá ligações com pontos interiores.

Golden et al [22] acrescentaram algumas modificações no algoritmo original de Clarke & Wright no sentido de torná-lo mais rápido e proporcionar uma economia de memória total de armazenamento de dados:

1)- Utilização do parâmetro 'γ' para definir savings modificados:

$$S_{ij} = d_{1i} + d_{1j} - \gamma\, d_{ij}$$

Wright no sentido de torná-lo mais rápido e

proporcionar uma economia de memória total de armazenamento de dados:

1)- Utilização do parâmetro 'γ' para definir savings modificados:

$$S_{ij} = d_{1i} + d_{1j} - \gamma\, d_{ij}$$

2)- Limitação do número de savings calculados. Golden sugere que se calcule savings somente para os nós que estejam 'próximos' entre si.

3)- Armazenamento dos savings S_{ij} em uma estrutura do tipo 'HEAP' para reduzir as operações de comparação e facilitar o acesso.

Vamos, agora, analisar, uma a uma, as modificações sugeridas por Golden et al:

a)- O parâmetro γ

O parâmetro γ foi inicialmente introduzido por Yellow [38], aparecendo, também, em um algoritmo desenvolvido por Gaskell [17]. O parâmetro γ tem grande influência sobre a geometria das rotas. Assim, quando γ cresce a partir do zero, o algoritmo tende a privilegiar os pares de nós mais próximos entre si, efetuando, primeiramente, as respectivas ligações. Por variar γ, o algoritmo oferece a oportunidade de se tomar uma solução melhor para um mesmo problema.

b)-Armazenamento

Golden et al sugerem duas mudanças principais no algoritmo original de Clarke & wright. A primeira diz respeito ao armazenamento das distâncias, d_{ij}, para os diversos pares de nós da rede. Em se tratando de um grafo não direcionado, cuja matriz de distâncias é simétrica, não é

necessário armazenar toda a matriz de distâncias e sim uma matriz triangular superior, a menos da diagonal principal que contém os elementos triviais ($d_{ij} = 0$). A segunda modificação proposta por Golden diz respeito à utilização e armazenagem dos savings. O mesmo autor sugere que não se faz necessário calcular todos os savings uma vez que muitos deles não serão utilizados pelo algoritmo que, em sua concepção geométrica, tende a ligar pontos próximos entre si. Golden, portanto, limita os savings, calculando-os somente para pontos 'vizinhos'. Golden define um critério de vizinhança por dividir a rede, dada por suas coordenadas retangulares x e y, em pequenos retângulos que somados compõem a área total da rede, permitindo, tão somente, os savings para os pares de nós pertencentes a retângulos vizinhos.

Nós programamos o algoritmo utilizando círculos para estabelecer o conceito de vizinhança e para isso, definimos o parâmetro ' R ' que estabelece o raio máximo de vizinhança permitida para os pontos, dois a dois, serem considerados para o cálculo de savings. Assim, para cada nó i, abrimos um círculo de raio R, centrado em i, e verificamos todos os nós que se encontram nesta vizinhança e, somente para estes, calculamos os savings. Uma vez definido o valor de R, compatível com a rede, o algoritmo trabalhará com uma lista menor de savings e, portanto, terá sua performance melhorada. Se armazenássemos os savings em uma matriz $S = (s_{ij})$ gastaríamos n^2 posições de memória. Utilizando a redução proposta por Golden, armazenamos as extremidades dos arcos, o valor do saving correspondente, em uma lista ordenada, gastando 3A posições de memória, onde A é o número de arcos que tiveram o saving calculado, considerando-se, novamente, a simetria dos savings ($S_{ij} = S_{ji}$).

c)-Estrutura de HEAP

Em cada passo do algoritmo estamos interessados em pegar o maior saving existente dentre todos os possíveis. Se o maior saving está associado a uma ligação viável, efetuamos a ligação correspondente e prosseguimos até que todos os nós estejam roteados. Golden sugeriu a utilização de uma estrutura de dados especial , o HEAP, para manipular fácil e rapidamente a lista de savings existente. Consideremos os savings $S_1, S_2, \ldots, S_m$ arranjados em uma estrutura do tipo árvore binária, que denominaremos HEAP. A propriedade fundamental do HEAP é a de que $S_i \geq S_{2i}$ e $S_i \geq S_{2i+1}$. A figura III.7 representa uma estrutura de HEAP, onde temos $S_1 > S_2$, $S_1 > S_3$, $S_2 > S_4$, $S_2 > S_5$, etc. A construção do HEAP é feita pela rotina CHEAP, que, no pior caso, tem uma complexidade computacional $O(m\log_2 m)$ com as comparações e trocas necessárias à construção. A regra de formação do HEAP coloca, evidentemente, o maior saving na posição S_1 no topo da lista. Vamos supor que a posição S_1 corresponda à ligação viável entre os nós i e j. Uma vez efetuada a ligação, colocamos o valor de S_1 em zero e 'empurramos' tal valor para o final do HEAP. Uma vez que estamos interessados, tão somente, em savings positivos, tal elemento não mais será considerado. Para 'empurrarmos' o elemento recém colocado em zero utilizamos a rotina ATHEAP, que atualizará o HEAP. A regra formadora da estrutura HEAP permite uma atualização bastante rápida, senão vejamos o exemplo da figura III.7:

fazendo $S_1 = 0$ (antes contendo o valor 68)
comparamos S_1 com o maior de seus filhos $S_3 = 67$ e teremos:
$S_1 = 67$ e $S_3 = 0$ (trocamos S_1 e S_3)
em seguida comparamos o novo valor de S_3 com o maior de seus filhos $S_7 = 58$ e teremos uma nova

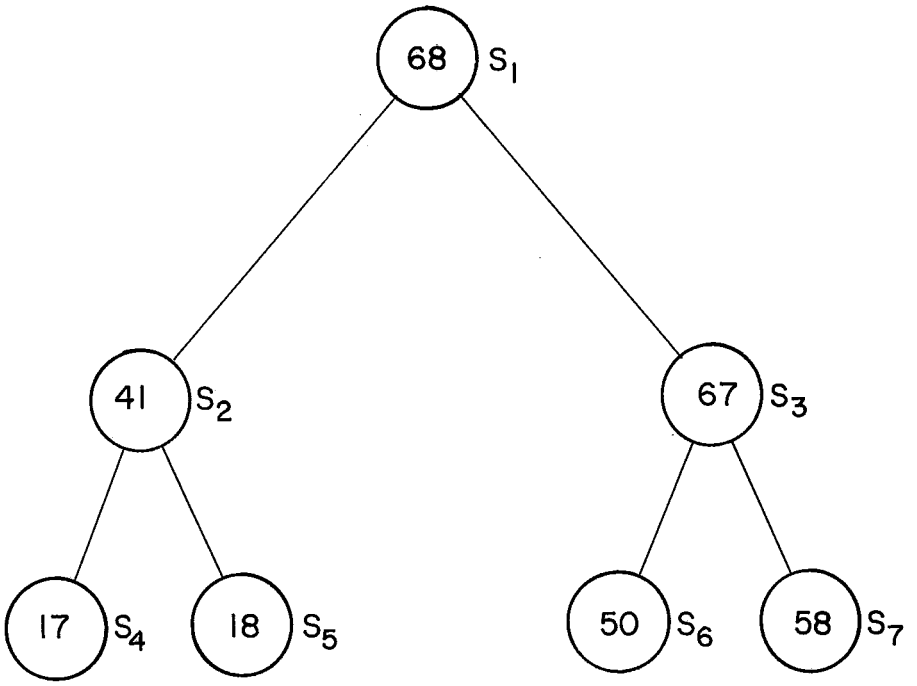


FIG. III.7 - ESTRUTURA DE "HEAP".

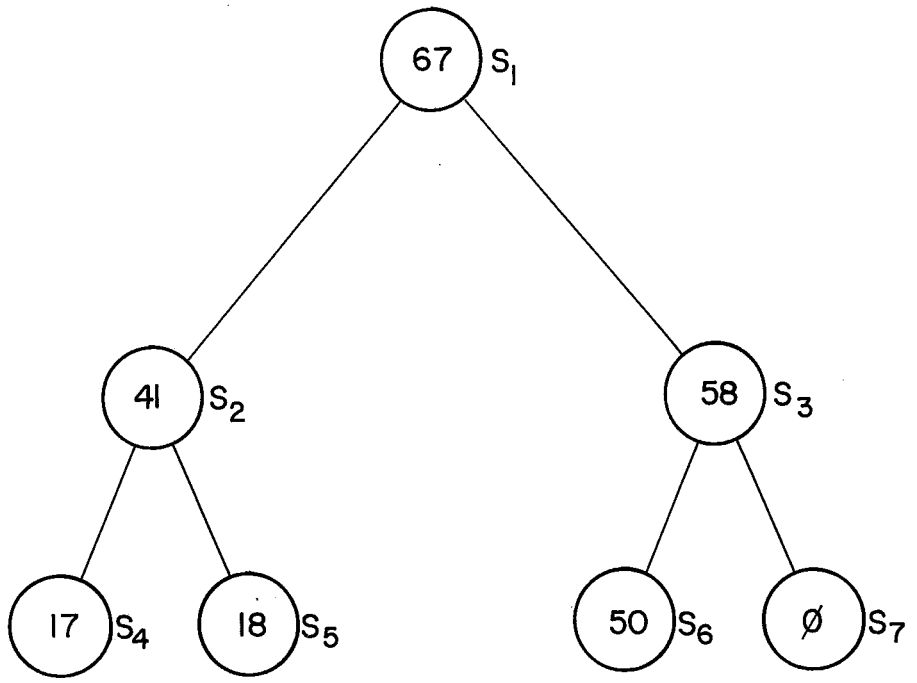


FIG. III. 8 - ATUALIZAÇÃO DO "HEAP".

troca:
$S_3 = 58$ e $S_7 = 0$ (trocamos S_3 e S_7)
chegamos, portanto, ao último nível do HEAP e temos, agora, a estrutura atualizada na figura III.8.
De forma mais geral o procedimento de atualização de um elemento na posição k do HEAP é feito comparando-se S_k com os seus filhos S_{2k} e S_{2k+1} até que $S_k \geq \max \{ S_{2k} , S_{2k+1}\}$ ou a posição S_k já está no último nível do HEAP. Uma observação se faz necessária, a de que devemos ter o HEAP completo em todos os seus níveis, ou seja, o primeiro nível, topo, com um elemento (2^0), o segundo nível com 2^1 elementos, o terceiro com 2^2 elementos e assim por diante. Isso é feito por completar com '- ∞ ' (qualquer valor menor que zero) as posições do último nível do HEAP, não preenchidas inicialmente com os savings.

Suponhamos, agora, que os nós i e j foram ligados sobre uma mesma rota e que i era um ponto já roteado e extremidade da rota r_1 e j foi aí acrescentado. Ora, evidentemente, o nó i transformou-se em um ponto interior e todas as entradas da lista remanescentes de savings que sugerem ligações com o nó i estarão inviabilizadas e, portanto, podem ser eliminadas, evitando a perda de tempo em um possível exame futuro pelo algoritmo. Uma vez que i transformou-se em ponto interior à rota r_1 percorremos a lista e para todas as posições que contenham ligação com o nó i aplicamos a rotina ATHEAP, colocando a referida posição em zero e 'empurrando' para baixo do HEAP. Maiores detalhes acerca da estrutura de HEAP podem ser vistos em Veloso [36].

O número de rotas a ser gerado pelo algoritmo tipo saving em paralelo não pode ser conhecido a priori. Podemos, entretanto, estabelecer mais um parâmetro, NMAXROT, que limita o número total de rotas a ser gerado pelo algoritmo. Um outro fato

importante é de que o algoritmo pode deixar alguns poucos nós sem ligação, ou melhor, no estado inicial em que se encontravam, ligados em uma rota exclusiva ao depósito. Tal fato pode tornar a solução final inviável uma vez que o número de rotas efetivamente gerado pelo algoritmo mais aquelas que permaneceram em seu estado inicial pode ser superior ao número máximo de rotas admissível. Em tal situação poderíamos, por exemplo, utilizar um algoritmo de inserção para colocar os poucos nós que sobraram. Evidentemente, poderíamos também experimentar uma mudança nos parâmetros γ, R ou NMAXROT.

Programamos o algoritmo de acordo com as modificações de Golden et al [22], programa VRPSAV, acrescentando a rotina de troca de arcos 2otm vista na seção III.2.1.b deste trabalho. Apresentamos a seguir uma bateria de testes com os parâmetros R e γ introduzidos por Golden. No primeiro teste nós geramos, aleatóriamente, dez problemas com 50 clientes cada e testamos a eficiência do parâmetro R através dos resultados listados na tabela III.9. As coordenadas retangulares, X e Y, de cada problema são geradas, respectivamente, nos intervalos [10,310] e [10,190], as coordenadas do depósito central, nó '1', são tomadas nos pontos médios dos intervalos correspondentes, os valores das cargas q_i são tomadas no intervalo [20,40]. As cargas e as coordenadas são geradas segundo uma distribuição uniforme de probabilidade de se tomar qualquer um dos valores nos respectivos intervalos. Na parte superior da tabela III.9 nós apresentamos os resultados sem a limitação do número de savings ($R = \infty$) e na inferior, os resultados com a limitação do número de savings (fazendo $R = 120$). A primeira coluna da referida tabela apresenta a distância total de roteamento, a segunda apresenta o número total de rotas geradas, a terceira o tempo total de processamento, em

R = ∞			
DISTANCIA	N.ROTAS	TEMPO(S)	N.SAVINGS
1822	5	14	1225
1952	6	14	1225
1760	5	14	1225
1901	5	14	1225
1854	5	14	1225
1942	6	13	1225
1938	5	14	1225
1834	5	14	1225
1848	6	14	1225
1966	6	14	1225
R = 120			
1822	5	8	652
1957	6	8	646
1760	5	8	677
1901	5	9	698
1820	6	8	663
1942	6	8	649
1952	6	8	629
1834	5	8	687
1848	6	8	639
1966	6	8	602

TABELA III.9

segundos, e a quarta e última coluna apresenta o número total de savings calculados.

Analisando os resultados contidos na tabela III.9, notamos que as distância obtidas com e sem a limitação do número de savings foram praticamente as mesmas, embora os tempos obtidos com a limitação

foram bem menores, cerca de 57 % . Agora, para os mesmos dez problemas vamos testar o parâmetro γ. Tomemos os valores 0.5, 1, 1.3 e 1.5 para γ e observemos as distâncias e o número de rotas obtidos em cada situação. Os problemas foram resolvidos fazendo R = ∞ e limitando o número de rotas ao máximo de seis. Os resultados do teste para o parâmetro γ encontram-se na tabela III.10. Observando os resultados da tabela III.10, percebemos que o aumento no valor de γ tende a gerar soluções melhores, podendo, entretanto, inviabilizar algumas delas, uma vez que o número de rotas ultrapassa a quantidade máxima permitida.

γ = 0.5	γ = 1	γ = 1.3	γ = 1.5
1872(5)	1822(5)	1803(6)	1709(6)
1983(5)	1952(6)	1965(6)	1979(6)
1919(6)	1760(5)	1784(6)	1794(7)*
1978(5)	1901(5)	1841(6)	1881(6)
1777(5)	1854(5)	1861(6)	1773(6)
2006(5)	1942(6)	1941(6)	1962(6)
1870(6)	1938(5)	1919(5)	1982(6)
1897(6)	1834(5)	1832(6)	1877(6)
1880(5)	1848(6)	1848(6)	1926(6)
1885(5)	1966(6)	2088(6)	2126(7)*

TABELA III.10- (os valores entre parênteses representam o número de rotas)
* soluções inviáveis quanto ao número de rotas

III.3.1.3 - Algoritmo sequencial de Mole & Jameson

R.H.Mole e S.R.Jameson [28] generalizaram o conceito de saving e criaram um algoritmo sequencial para o PRV1. No algoritmo anterior os savings eram permitidos, tão somente, para os pontos extremos das rotas, ou seja, permitia-se o saving entre dois nós extremos de rotas distintas ou o saving entre um nó extremo de uma determinada rota e um outro nó ainda não roteado. O algoritmo, de forma semelhante ao de Clarke & Wright, parte da situação incial onde cada cliente é sevido de forma exclusiva por um único veículo. Vejamos, agora, a generalização do saving:

$$SAV_C(A,B) = 2\ d_{OC} + (d_{AB} - d_{AC} - d_{BC}),$$

onde A e B são nós situados sobre uma mesma rota. O representa a origem (depósito central) e C é um nó ainda não roteado.

O saving representa a economia obtida quando incluimos o nó C, ainda não roteado, entre os nós A e B. O termo entre parênteses representa o esforço de inclusão do nó C entre os nós A e B. De maneira equivalente temos:

$$ESF_C(A,B) = d_{AC} + d_{BC} - d_{AB}$$

$$SAV_C(A,B) = 2\ d_{OC} - ESF_C(A,B)$$

Mole & Jameson definem, ainda, os savings modificados:

$$MESF_C(A,B) = d_{AC} + d_{BC} - \mu.d_{AB} \qquad \text{(III.17)}$$

$$MSAV_C(A,B) = \lambda\, d_{OC} - MESF_C(A,B) \qquad \text{(III.18)}$$

Na expressão III.17 foi introduzido o parâmetro μ relacionado com a geometria da rota que se pretende construir definindo, portanto, o esforço modificado MESF. Na expressão III.18 introduziu-se o parâmetro λ para definir o saving modificado MSAV. O parâmetro λ está associado à distância entre os diversos nós, clientes, e o depósito, enquanto o parâmetro μ está associado às distâncias entre os clientes. Por manipular os parâmetros podemos tentar obter soluções melhores do que os obtidos com o saving em sua forma original. A atribuição de valores maiores para λ faz com que o algoritmo tenda a ligar, primeiramente, pontos mais afastados do depósito, enquanto a atribuição de valores maiores para μ tende a privilegiar ligações entre nós próximos entre si. Mole & Jameson [28] citam, entre outras, as duas combinações dos parâmetros:

λ	μ		CRITÉRIO
2	1	==>	Saving Generalizado de Clarke & Wright
0	1	==>	Minimo "esforço"

O algoritmo de Mole & Jameson inicia a construção de uma rota tomando o cliente mais distante do depósito e, ainda, não roteado. Em seguida verifica, para todo nó ainda não roteado e passível de ser incluído na rota, a melhor posição de inclusão na rota utilizando o critério do menor esforço modificado (MESF). Nesta etapa o algoritmo anotaria, portanto, o valor do MESF e a posição de inclusão na rota. O algoritmo utiliza em seguida o critério de máximo saving modificado (MSAV) para

selecionar o nó a ser incluído na rota e na posição correspondente ao seu MESF. Ambos os procedimentos são representados pelas expressões:

$MESF_C$ (I,J) = min { $MESF_C$ (A,B) } (III.19)

para todo par de nós adjascentes (A,B) sobre a rota;

$MSAV_C$ (L,M) = max { $MSAV_C$ (I,J) } (III.20)

para todo nó C ainda não roteado e viável;

Após cada inclusão, o algoritmo utiliza a rotina 2otm (seção III.2.1.b), para tentar um melhoramento na rota ainda em construção. Uma vez que tenha sido melhorada pela rotina 2otm e, por conseguinte, a posição relativa dos nós tenha sido modificada o algoritmo calcula novamente os MESF, anotando a melhor posição de inclusão, para todos os nós ainda não roteados e viáveis. Se a rotina 2otm não promoveu nenhuma alteração na rota (não melhorou), o algoritmo passa a atualizar os MESF ao invés de recalculá-los. Seja C o último nó incluído na rota na posição (L,M), ou seja, entre os nós L e M. Para todo nó X ainda não roteado e viável o algoritmo compara o seu MESF anterior com as duas novas possibilidades de inclusão sobre a rota, as posições (L,C) e (C,M) tomando o menor valor para o MESF atualizado. O procedimento é sem dúvida muito mais rápido do que o cálculo de todos os MESF. O algoritmo repete o procedimento até que a rota não possa mais ser expandida, partindo, então para a inicialização de uma nova rota. O término se verifica quando todos os nós estão roteados.

Programamos o algoritmo de Mole & Jameson em sua versão original, programa MOLEJAME, e rodamos os mesmos dez problemas de roteamento apresentados anteriormente nas tabelas III.9 e III.10. Tomamos quatro combinações distintas de parâmetros e listamos os resultados na tabela III.11.

$\lambda = 0$ $\mu = 1$	$\lambda = 1.5$ $\mu = 2.5$	$\lambda = 2$ $\mu = 1$	$\lambda = 2$ $\mu = 2.5$
2211(5)* [18]**	1868(5) [19]	1918(5) [18]	1775(5) [19]
2255(5) [18]	2111(5) [21]	2022(5) [16]	2069(5) [18]
2122(6) [16]	1977(5) [23]	1994(5) [16]	1982(5) [20]
2045(5) [17]	1924(5) [21]	2009(5) [19]	1878(5) [21]
2019(5) [19]	1855(5) [20]	2016(5) [18]	1940(5) [20]
2040(5) [15]	2027(5) [19]	1957(5) [16]	2119(5) [17]
2098(5) [18]	2051(5) [21]	1972(5) [17]	2066(5) [19]
2242(5) [17]	2168(6) [23]	1965(5) [20]	1941(6) [21]
2218(5) [18]	1858(5) [21]	2020(5) [20]	2046(5) [22]
2204(5) [18]	2066(5) [21]	2057(5) [16]	2009(5) [18]

Tabela III.11 - Resultados computacionais para o algoritmo sequencial de Mole & Jameson. Resultados obtidos em um microcomputador PC-XT.
* número de rotas gerado pelo algoritmo
** tempo de processamento em segundos

A análise da tabela III.11 revela uma ligeira superioridade dos resultados obtidos na última coluna, ou seja, para a combinação $\lambda = 2$ e $\mu = 2.5$. Para esta combinação o algoritmo estaria trabalhando, de forma semelhante ao algoritmo de Clarke & Wright, tendendo a ligar, primeiramente, os nós mais distantes do depósito, dando, entretanto, maior ênfase àquelas ligações de pontos próximos entre si.

É interessante observar que um algoritmo sequencial como o de Mole & Jameson tende a aproveitar ao máximo a capacidade de cada rota, ou seja, colocar o máximo de clientes possíveis,

fornecendo, em geral, um menor número de rotas do que o algoritmo de Clarke & Wright, como pode ser visto nos quadros III.9, III.10 e III.11. Tal procedimento pode prejudicar a qualidade da solução obtida uma vez que os últimos nós incluídos em uma determinada rota podem ter um MESF elevado e, então, seria melhor que fossem incluídos em uma outra rota, não violando, entretanto, a disponibilidade de veículos da frota. Mole & Jameson sugerem algum tipo de refinamento 'pós-processamento' para evitar esta tendência do algoritmo.

Uma vantagem evidente de um algoritmo sequencial sobre outro que gere rotas em paralelo é a maior facilidade que o primeiro apresenta de manipular mais facilmente restrições individuais sobre cada rota (ou cada veículo), como por exemplo restrições de capacidade de carga máxima, tipo diferenciado de compartimentos para carga e até mesmo restrições relativas à habilidade profissional dos condutores dos veículos.

III.3.2 - Problema de Roteamento de veículos caso multi-depósito (PRVM)

Uma extensão do problema de Roteamento de Veículos é aquela em que os veículos estão estacionados em diferentes depósitos, vinculados a estes, tendo a missão de visitar um conjunto de clientes cuja demanda e localização são conhecidas. Os veículos têm uma capacidade máxima de carga que deve ser respeitada e os clientes devem ser visitados por um único veículo que lhes atenda a demanda. Este é o Problema Básico de Roteamento de Veículos Multi-depósito (PRVM) que passaremos a descrever.

III.3.2.1 - Modelagem

$$\text{Min} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{NV} d_{ij} X_{ij}^{k} \qquad \text{(III.21)}$$

S.A.

$$\sum_{i=1}^{n} \sum_{k=1}^{NV} X_{ij}^{k} = 1 \qquad (j=M+1,\ldots,N) \qquad \text{(III.22)}$$

$$\sum_{j=1}^{n} \sum_{k=1}^{NV} X_{ij}^{k} = 1 \qquad (i=M+1,\ldots,N) \qquad \text{(III.23)}$$

$$\sum_{i=1}^{n} X_{ij}^{k} - \sum_{j=1}^{n} X_{ij}^{k} = 0 \quad (k=1,\ldots,NV)\,(p=1,\ldots,n) \qquad \text{(III.24)}$$

$$\sum_{i=1}^{n} q_i \sum_{j=1}^{n} X_{ij}^{k} \leq \text{cap}(k) \quad (k=1,\ldots,NV) \qquad \text{(III.25)}$$

$$\sum_{i=1}^{M} \sum_{j=M+1}^{n} X_{ij}^{k} \leq 1 \quad (k=1,\ldots,NV) \qquad \text{(III.26)}$$

$$\sum_{p=1}^{M} \sum_{i=M+1}^{n} X_{ip}^{k} \leq 1 \quad (k=1,\ldots,NV) \qquad \text{(III.27)}$$

$X \in S$ (III.28)

$X_{ij}^{k} = 0$ ou 1 (III.29)

onde

V = conjunto dos nós {1,...,N}
M = número de depósitos (localizados em {1,...,M})
NV = número total de veículos disponíveis
cap(k) = capacidade do veículo k.
q_i = demanda do cliente i (q_i=0 para i= 1,...,M)
d_{ij} = distância ou custo entre os nós i e j

$$x_{ij}^{k} = \begin{cases} 1, \text{ se } (i,j) \text{ está sobre a rota de } k \\ 0, \text{ caso contrário} \end{cases}$$

$X = (x_{ij}^{k})$ Vetor representando as ligações

S = conjunto de todos os subciclos possíveis contendo um e somente um elemento de {1,2,...,M}

A função objetivo (III.21) representa a minimização das distâncias totais (ou custos) percorridos por todos os veículos. As restrições (III.22) e (III.23) indicam que cada cliente será servido por um único veículo. As restrições (III.24) obriga cada veículo a entrar e sair de um mesmo nó (uma vez que tenha entrado). As restrições (III.25) garantem que o veículo transportará uma carga menor ou igual a sua capacidade máxima. As restrições (III.26) e (III.27) garantem que cada veículo irá efetuar no máximo uma rota. A restrição (III.28) garante a eliminação de subciclos que não contenham um depósito. As restrições (III.29) definem as componentes da variável X como sendo do tipo 0 ou 1.

III.3.2.2. Algoritmos para o PRVM

Ao contrário do PRV1, encontramos poucos algoritmos eficientes para o PRVM. A tentativa de extender alguns algoritmos originalmente concebidos para o PRV1 não é imediata. Vimos na seção anterior que os algoritmos de Clarke & Wright e o de Mole & Jameson são concebidos sob 'inspiração ' geométrica inerente ao PRV1. Uma vez que o PRVM tem uma 'geometria' diferente, a extênsão imediata daqueles algoritmos constituiria numa péssima estratégia para o problema multi-depósito.

Citamos os algoritmos de Golden et al [22], Wren & Holliday [37], Gillett & Johnson [19] e Cassidy & Bennett [7] como sendo algoritmos heurísticos eficientes para resolver o PRVM. Os algoritmos de Wren & Holliday e Cassidy & Bennett utilizam a mesma estratégia básica, partem de uma solução inicial qualquer para efetuar em seguida uma série de refinamentos inter e intra-rotas, com a troca de um ou mais arcos em cada iteração. Em geral, o algoritmo termina quando nenhum outro melhoramento pode ser conseguido com as rotinas de refinamento, situação em que a solução teria convergido para um ótimo local. Wren & Holliday [37] citam a possibilidade de adotar um critério diferente, limitando a atuação do algoritmo a um tempo máximo de processamento. É interessante observar, também, que estes algoritmos permitem uma fácil e, muitas vezes, desejável iteração homem/máquina, quando o processamento pode ser interrompido e a solução corrente refinada pelo controlador do sistema, de maneira visual.

De outro lado temos os algoritmos de Gillett & Johnson [19] e Golden et al [22]. O primeiro é uma extensão do algoritmo 'sweep' concebido por Gillett & Miller [18] para o PRV1. O algoritmo consiste basicamente de duas fases: na primeira utiliza-se um critério para agrupar os nós ao redor

dos depósitos e na segunda fase utiliza-se o algoritmo 'sweep' para rotear os nós já associados aos depósitos. O segundo algoritmo, apresentado por Golden et al, é uma extensão do método de 'saving' para o PRVM e será motivo da nossa atenção a seguir.

III.3.2.2.1 - Algoritmo tipo Saving para o PRVM

Golden et al [22] apresentam uma versão modificada do algoritmo de Tillman & Cain [35], sendo este último uma extensão do algoritmo de Clarke & Wright para o PRVM. O algoritmo parte de uma solução inicial trivial, onde todos os clientes são atendidos por um veículo situado no depósito mais próximo. Seja,

d_{ij} = distância entre os clientes i e j

d_i^k = distância entre o nó i e o depósito k

A distância total da solução inicial é dada pela expressão:

$$D = \sum_{i=1}^{N} 2 \min_{k} \{ d_i^k \}$$

onde N é o número total de clientes.

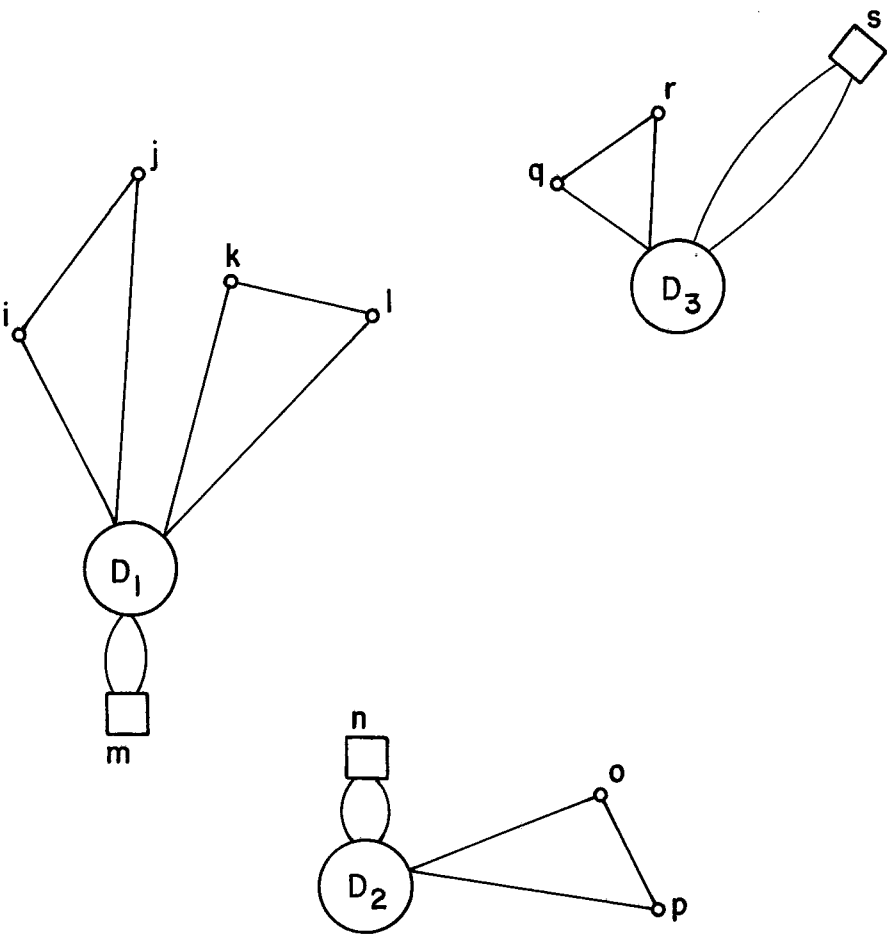
O algoritmo utilizando a mesma idéia do algoritmo de Clarke & Wright liga em cada iteração um par de nós no sentido de decrescer a distância total percorrida. Os savings são calculados entre dois nós quaisquer, i e j, e um depósito, k, de forma diferente do saving para o PRV1:

$$s_{ij}^{k} = \bar{d}_{i}^{k} + \bar{d}_{j}^{k} - d_{ij}$$

onde

$$\bar{d}_{i}^{k} = \begin{cases} 2 \min_{t} \{ d_{i}^{t} \} - d_{i}^{k} , & \text{se i ainda não foi associado definitivamente a um depósito.} \\ d_{i}^{k} , & \text{no caso contrário.} \end{cases}$$

No primeiro caso temos a situação em que o nó i ainda não foi colocado sobre nenhuma rota e, portanto, ainda está ligado a seu depósito mais próximo, numa situação que não é,ainda, definitiva. No segundo caso o nó i já está sobre uma rota associada ao depósito k e o saving é calculado de forma equivalente ao saving para o PRV1. A figura III.12 ilustra os dois estados possíveis de ligação. Na figura, por exemplo, a ligação entre os nós m e n sobre uma rota, pertencente ao depósito D_2 , seria enquadrada no primeiro critério. Por outro lado, a ligação entre os nós j e k, já designados para o depósito D_1 se enquadraria no segundo critério. Uma vez que um determinado nó i tenha sido colocado sobre uma rota associada ao depósito k, o algoritmo não permite que ele seja transferido de depósito. Nota-se, portanto, que toda vez que um nó i é designado para um depósito, k, torna-se necessário atualizar todos os demais savings que contenham o elemento i, eliminando, inclusive, os savings referentes a outros depósitos que não o depósito k. O algoritmo, em cada iteração, seleciona o maior saving e, se viável, efetua a ligação correspondente para em seguida promover a atualização de alguns destes savings, conforme



D_i depósito i

o cliente já ligado sobre rota e, portanto, definitivamente associado a um depósito.

□ cliente ainda não ligado definitivamente a um depósito.

FIG. III.12 - SAVING MULTI - DEPÓSITO.

explicado anteriormente. O procedimento se repete até que todos os nós tenham sido roteados ou a pilha de savings tenha sido esgotada.

Uma questão fundamental neste algoritmo é a forma como se vai armazenar a imensa quantidade de savings e, claro, uma forma eficiente de manipulá-los. Em cada passo do algoritmo temos que selecionar o maior saving entre uma quantidade muito grande de possibilidades e, subsequentemente, temos que atualizar uma certa quantidade destes savings. Golden et al [22] sugerem, de forma semelhante ao algoritmo tipo saving para o PRV1, uma limitação do número de savings e, principalmente, a utilização de estrutura de dados que explore a simetria destes, no sentido de economizar memória e agilizar o algoritmo.

Golden afirma que o presente algoritmo não é capaz de tratar eficientemente problemas maiores, com cerca de mil nós, uma vez que a quantidade de memória necessária e o tempo de processamento seriam demasiadamente elevados. Tomando, por exemplo, um problema com 1000 nós e 100 depósitos, considerando os valores dos savings como sendo inteiros, gastaríamos aproximadamente 100 Mb (cem Megabytes) de memória para armazenar todos os savings, isto já considerando a simetria dos savings.

Golden et al [22] sugerem uma modificação, o que seria uma segunda versão para o mesmo algoritmo, que seria eficiente para manipular problemas de maior magnitude. Para tanto, Golden utilizou uma estratégia originalmente colocada por Gillett & Johnson [19]:

seja $r(i) = d_i^{k_1}/d_i^{k_2}$ a razão entre as distâncias do nó i ao seu primeiro e segundo depósitos mais próximos, respectivamente, e δ um parâmetro tomado no intervalo [0,1]. O parâmetro δ é utilizado para definir duas classes de nós, os nós para os

quais $r(i) \geq \delta$, aos quais chamaremos nós do tipo I e aqueles em que $r(i) < \delta$, nós do tipo II. Inicialmente, associamos os nós do tipo II ao depósito k_1 correspondente, ou seja, ao seu depósito mais próximo. Em seguida aplicamos o algoritmo de saving multi-depósito para aqueles nós do tipo I. No final destes dois procedimentos teríamos alguns nós ligados sobre rotas associadas aos depósitos e um grupo de nós associados aos depósitos, porém, ainda não roteados. Numa segunda fase aplicaríamos o algoritmo de saving para PRV1 tantas vezes quantos fossem os depósitos até que todos os nós tenham sido roteados. Por manipular o parâmetro δ poderíamos tomar diferentes soluções e tomar a melhor dentre todas as obtidas. A capacidade do algoritmo em manipular um maior número de nós do tipo I definiria a sua maior ou menor eficiência. Assim, quando tomamos um valor menor para o parâmetro δ estamos, em verdade, considerando um maior número de nós do tipo I e, em contrapartida, exigindo um maior esforço do algoritmo multi-depósito.

CAPÍTULO IV

O PROBLEMA DOS AUDITORES

IV.1 - Enunciado do Problema

No primeiro capítulo deste trabalho, a título de introdução, fizemos uma breve descrição do Problema de Roteamento de Auditores, doravante designado por PRA, não revelando todos os detalhes que cercam o problema em toda a sua extensão, o que será feito a seguir.

Um grupo de auditores, inicialmente localizados em suas praças-sedes, têm a missão de visitar, dentro de um período limitado de tempo, algumas das diversas agências do Banco do Brasil S.A, selecionadas para serem auditadas no referido período. Cada auditor realiza, portanto, uma única rota, que começa e termina na sua praça-sede. O auditor permanece em cada agência por um período de tempo suficiente para a realização de seu trabalho na referida agência. O tamanho deste período de tempo, medido em dias, é determinado a priori e é função do porte da agência e, também, função do desempenho da mesma. Algumas poucas agências, de maior porte, requerem a visita simultânea de dois auditores, situação que designaremos por missão conjunta. Uma outra restrição que se impõe sobre o roteamento de auditores é que alguns auditores são impedidos de visitarem algumas das agências. O impedimento está relacionado com questão de segurança dos serviços e são, basicamente, dois os motivos que geram o impedimento:

a)- O fato de existir parentesco entre o auditor e a administração da agência;

b)- O fato de o auditor ter sido o último auditor a

realizar serviços na agência;

O meio de transporte utilizado pelos auditores é, em geral, o rodoviário, não estando impedido, entretanto, a utilização de um outro meio quando a situação exige. Em geral o tempo de viagem entre uma e outra agência ou entre sua praça-sede e uma agência não ultrapassa o período de um dia.

Diante das restrições apontadas anteriormente estamos interessados em encontrar uma configuração de rotas que minimize a distância total percorrida por todos os auditores dentro do período de planejamento das rotas. Pode-se notar a semelhança deste problema com o problema de Roteamento de veículos multi-depósito, PRVM. Uma analogia entre um e outro problema poderia ser representada da seguinte maneira:

PRA		PRVM
Auditor	<———>	Veículo
Praça-sede	<———>	depósito
tempo de visita	<———>	carga

A diferenciar os dois problemas, teríamos o fato de no PRA aparecerem as missões conjuntas, onde dois auditores visitam, simultâneamente, o mesmo nó (agência).

O PRA apresenta-se com a configuração de até duzentos auditores situados em praças-sedes distintas e até mil agências candidatas a serem auditadas dentro de um período de planejamento que, em geral, é considerado como sendo de dois meses. Temos, portanto, um problema de grande porte e de forma semelhante ao PRVM não pode, ainda, ser resolvido de maneira exata. Em seguida apresentaremos a modelagem do PRA pela sua formulação em Programação Matemática e um algoritmo heurístico para a sua resolução.

IV.2 - Modelagem

$$\text{Min} \quad \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{NA} d_{ij} \, X_{ij}^{k} \qquad \text{(IV.1)}$$

s.a.

$$\sum_{i=1}^{N} \sum_{k=1}^{NA} X_{ij}^{k} = 1 \qquad (j=M+1,\ldots,M+NAS) \qquad \text{(IV.2)}$$

$$\sum_{j=1}^{N} \sum_{k=1}^{NA} X_{ij}^{k} = 1 \qquad (i=M+1,\ldots,M+NAS) \qquad \text{(IV.3)}$$

$$\sum_{i=1}^{N} \sum_{k=1}^{NA} X_{ij}^{k} = 2 \qquad (j=M+NAS+1,\ldots,M+NAS+NAC) \qquad \text{(IV.4)}$$

$$\sum_{j=1}^{N} \sum_{k=1}^{NA} X_{ij}^{k} = 2 \qquad (i=M+NAS+1,\ldots,M+NAS+NAC) \qquad \text{(IV.5)}$$

$$\text{Se } X_{ij}^{k_1} = 1 \text{ e } X_{lj}^{k_2} = 1 \Longrightarrow T_{k_1}^{j} = T_{k_2}^{j}$$
$$(i,l=1,\ldots,N) \quad (j=N+NAS+1,\ldots,N) \qquad \text{(IV.6)}$$

$$\sum_{i=1}^{N} X_{ip}^{k} - \sum_{j=1}^{n} X_{pj}^{k} = 0 \qquad (k=1,\ldots,NA) \quad (p=1,\ldots,N) \qquad \text{(IV.7)}$$

$$\sum_{i=1}^{N} t_i \left(\sum_{j=1}^{N} X_{ij}^{k} \right) \leq TMAXROT(K) \qquad (k=1,\ldots,NA) \qquad \text{(IV.8)}$$

$$\sum_{i=1}^{M} \sum_{j=M+1}^{N} X_{ij}^{k} \leq 1 \qquad (k=1,\ldots,NA) \qquad \text{(IV.9)}$$

$$\sum_{p=1}^{M} \sum_{i=M+1}^{N} X_{ip}^{k} \leq 1 \qquad (k=1,\ldots,NA) \qquad \text{(IV.10)}$$

$$X_{ij}^{k} = 0 \text{ se } IMP_{k}^{i} = 1 \text{ ou } IMP_{k}^{j} = 1$$
$$(i,j=1,2,\ldots,N) \quad (k=1,\ldots,NA) \qquad \text{(IV.11)}$$

$$X \in S \qquad \text{(IV.12)}$$

$$X_{ij}^{k} = 0 \text{ ou } 1 \qquad \text{(IV.13)}$$

onde

d_{ij} = distância entre dois nós i e j
V = conjunto de todos os nós {1,...,N}
N = número total de nós
NA = número de auditores disponíveis
M = número de praças-sedes representadas pelos nós {1,...,M}
NAS = número de agências de missão simples, representado pelos nós {M+1,...,M+NAS}
NAC = número de agências de missão conjunta, representado pelos nós {M+NAS+1,...,M+NAS+NAC}
t_i = tempo requerido para auditar a agência i
TMAXROT(K) =tempo máximo de rota permitidoao auditor k.
T_k^j = tempo de chegada do auditor k no nó j

$$IMP_k^i = \begin{cases} 0, \text{ se não exite impedimento para o k-ésimo auditor visitar a agencia i.} \\ 1, \text{ caso contrário} \end{cases}$$

S = conjunto de todos os subciclos viáveis

$$X_{ij}^{k} = \begin{cases} 1, \text{ se o arco (i,j) está sobre a rota do k-ésimo auditor} \\ 0, \text{ caso contrário} \end{cases}$$

A expressão (IV.1) traduz o fato de querermos minimizar a distância total envolvida no processo de roteamento. As restrições (IV.2) asseguram a entrada de um único auditor em qualquer um dos nós que representam agencias que requerem missão

simples. As restrições (IV.3) asseguram a saída de um único auditor de qualquer um dos mesmos nós. As restrições (IV.4) e (IV.5) asseguram a entrada e saída, respectivamente, de dois auditores para aquelas agências que exigem missão conjunta. As restrições (IV.6) asseguram a simultaneidade da chegada dos auditores que fazem a missão conjunta. Temos, portanto, esta restrição controlando o 'scheduling' dos auditores que fazem missão conjunta. As restrições (IV.7) garantem a continuidade das rotas para todos os auditores, ou seja, um auditor que tenha entrado em algum nó deverá, também, sair do respectivo nó. As restrições (IV.8) limitam o tempo de rota para os auditores. Aqui uma observação se faz necessária: nós desprezamos os tempos de viagem dos auditores, considerando, tão somente, os tempos t_i de permanência nas agências. As restrições (IV.9) e (IV.10), em conjunto, asseguram que cada auditor não efetuará mais do que uma rota. As restrições (IV.11) asseguram que nenhum auditor visitará agência para o qual esteja impedido. A função de impedimento IMP_k^i é, evidentemente, conhecida a priori. A restrição (IV.12) elimina a possibilidade de formação de subciclos inviáveis, ou seja, subciclos que não contenham uma praça-sede. E, finalmente, a restrição (IV.13) define a variável X_{ij}^k como sendo booleana.

É interessante observar a semelhança entre a presente formulação do PRA e a formulação do PRVM fornecida no capítulo anterior. A diferença principal reside no fato de que, no PRA, admitimos a existência de dois arcos incidindo em um mesmo nó, configurando, assim, missão conjunta. Na prática, o número de agências que exigem missões conjuntas é bem inferior ao número de agências de missão simples. Outra diferença entre os dois referidos problemas é quanto às restrições de impedimentos que aparecem no PRA. A situação

prática apresenta, em geral, um máximo de cinco impedimentos por auditor. Vemos, portanto, que em um problema de porte, por exemplo, duzentos auditores e mil agencias, o número de arcos impedidos é ínfimo quando comparado com a totalidade das ligações possíveis.

A forma como modelamos o PRA pode ser identificada com a modelagem do PRVM para concluirmos que o primeiro não tem resolução, de forma exata, menos difícil do que o segundo que, como vimos no capítulo anterior, encontra-se no campo dos problemas NP-completos. Uma particularidade do PRA que, a nosso ver, o torna mais difícil do que os problemas de roteamento que aparecem na literatura é o excessivo número de praças-sedes, uma vez que os cerca de duzentos e cinquenta auditores do Banco do Brasil acham-se espalhados por diferentes cidades deste País. Em geral, percebemos que os problemas de roteamento, ou assemelhados, que aparecem na literatura tratam de problemas onde o número de depósitos (ou análogo) não ganha tal ordem de magnitude.

IV.3 - Resolução do PRA

IV.3.1 - Elementos de Análise de Grupamentos

Antes de apresentarmos o nosso algoritmo heurístico para o PRA, faz-se necessário apresentar alguns conceitos básicos de Análise de grupamento que nos serão de grande valia quando da apresentação do algoritmo.

Consideremos um conjunto de n elementos $E = \{e_1, e_2, \ldots, e_n\}$ dotados, cada um deles, de p características observáveis e mensuráveis.

Denotando as medidas das características do elemento e_j pelo vetor coluna $X_j = [X_{1j},...,X_{pj}]^t$, é possível resumir a descrição dos n elementos da população em um conjunto de vetores $X = \{ X_1, X_2,...X_n\}$ representados no R^p.

O problema de Análise de Grupamento pode ser resumido na seguinte afirmação: "Com base no conjunto X, determinar uma partição P_m dos objetos pertencente a E em $\underline{m}$ grupos $g_1, g_2,...,g_m$, alocando cada e_j a apenas um grupo, de forma a que elementos semelhantes sejam alocados a grupos distintos" [27]. A solução de um problema de Análise de Grupamento pode, portanto, ser encarada como uma partição do conjunto E que otimize uma função objetivo $f(P_m)$, função essa que reflita uma medida quantitativa de semelhança 'intra' e 'entre' grupos.

IV.3.1.1 Medidas de dispersão interna de um Grupo

Vimos que o objetivo da Análise de Grupamento é reunir os elementos em grupos semelhantes. O conceito de semelhança deve, portanto, ser quantificado para que possa receber um tratamento matemático. Uma forma de se medir tal semelhança é medir a dispersão interna dos elementos dentro de um grupo. Lucas [27], cita as principais medidas de dispersão utilizadas:

- Soma dos quadrados dentro do grupo
- Variância interna de grupo
- Diâmetro de grupo
- Dispersão via mediana de grupo

Nos interessará a primeira forma, ou seja, a Soma dos quadrados dentro do grupo que passaremos a descrever.

Consideremos um conjunto E de elementos que desejamos particionar em $\underline{m}$ grupos g_1, $g_2,\ldots,g_m$, de forma a reunir os elementos mais semelhantes dentro de cada grupo. Definiremos, portanto, uma função objetivo que traduza matematicamente tal semelhança e para isso, utilizaremos o conceito de 'soma dos quadrados dentro do grupo'. Seja

$$W_I = \sum_{i=1}^{n_I} (X_i - \bar{X}_I)^t (X_i - \bar{X}_I)$$

$$\text{onde } \bar{X}_I = \frac{1}{n_I} \sum_{i=1}^{n_I} X_i$$

e $\bar{X}_I$ é dita a média ou centróide de grupo, n_I é o número de elementos do grupo g_i.
Assim, W_I representa a soma dos quadrados das distâncias euclideanas entre cada elemento do grupo e a centróide do mesmo. Poderíamos, portanto tomar para função objetivo $f(P_m) = \sum_{i=1}^{m} W_I$, resolvendo um problema de minimização. Teríamos, por conseguinte, a seguinte situação:

Minimizar $f(P_m)$

sujeito a P_m viável

IV.3.1.2 - Métodos de Análise de Grupamento

Lucas [27] menciona em seu trabalho sobre a enorme quantidade de técnicas existentes em Análise de Grupamento, citando que boa parte destas podem ser encontrados nos textos de Andeberg [2], Diday e Simon [13], Duran e Odell [15], Hartigan

[23] e Pinho Gama [31]. Lucas classifica os métodos de Análise de Grupamento em três famílias:

- hierarquizados aglomerativos (HA)
- de realocação iterativa (RI)
- de programação matemática (PM)

Os métodos chamados hierarquizados aglomerativos são inicializados tomando-se a partição com n grupos $\{e_1\}$, $\{e_2\},\ldots,\{e_n\}$ do conjunto $E = \{e_1, e_2,\ldots,e_n\}$. O processo iterativo consiste em se ligar a cada vez dois dos grupos, considerados como sendo os mais 'semelhantes', resultando portanto na redução de um grupo a cada iteração. O algoritmo termina quando se tem o número de grupos desejado ou, evidentemente, quando obtemos um único grupamento E. A 'semelhança' entre dois grupos deve ser quantificada de maneira que possamos dar um tratamento matemático ao problema. Define-se, para tanto, o conceito de distância inter-grupos. Várias são as maneiras de se tomar a distância entre dois grupos (d^g_{ij}):

a)- método da ligação simples

$$d^g_{ij} = \min_{\substack{e_i \in g_I \\ e_j \in g_J}} d_{ij}$$

onde d_{ij} mede a distância entre os elementos i e j situados, respectivamente, nos grupos g_I e g_J. Para se medir a distância entre os elementos podemos usar qualquer métrica, por exemplo, a euclideana.

b) método da centróide

$$d_{ij}^{g} = d_{ij}^{2} = d_{2}^{2} (\bar{X}_I , \bar{X}_J),$$

aonde $\bar{X}_I$ e $\bar{X}_J$ são, respectivamente, as médias, ou centróides, dos grupos g_I e g_J, ou seja,

$$X_I = \frac{1}{n_I} \sum_{i=1}^{n_I} X_i$$

utilizamos a métrica euclideana, d_2, para medir a distância entre as centróides dos grupos.

É importante distinguir entre distância entre elementos e distância entre grupos. Mostramos anteriormente duas maneiras de se medir a distância entre dois grupos, mas existem outras formas de se medir tais distâncias, como pode ser visto em Lucas [27]. Por fim, a respeito dos métodos hirearquizados aglomerativos, gostaríamos de ressaltar a sua natureza heurística, não existindo, portanto, nenhuma garantia de se chegar ao ótimo.

Uma forma de se obter a partição ótima de determinado conjunto E, com n elementos, por exemplo, em m grupamentos seria a enumeração de todas as alternativas possíveis de partições P_m, calculando-se o valor da função objetivo $f(P_m)$ para cada uma delas e escolhendo-se a melhor P_m^*. Entretanto, o número S(n,m) de alternativas possíveis de se particionar E em m grupos é extremamente elevado, segundo Duran e Odell [15], dado por:

$$S(n,m) = \frac{1}{m!} \sum_{k=0}^{m} \binom{m}{k} (-1)^{m-k} K^n$$

O problema é de natureza combinatorial e, portanto, não se mostra prático tentar resolvê-lo

por enumeração. Tal fato fez com que alguns pesquisadores desenvolvessem uma série de algoritmos exatos, baseados em Programação Dinâmica, Teoria de Grafos e Programação Inteira. Tais algoritmos, entretanto, conseguem manipular de forma eficiente, tão somente, problemas de pequeno porte. Tais algoritmos compõem a terceira classe definida por Lucas [27]. A seguir dedicaremos uma seção especial à classe dos métodos de realocação iterativa que, efetivamente, servirão de base para o algoritmo que iremos desenvolver para o Problema dos Auditores.

IV.3.1.2.1 - Métodos de Realocação Iterativa

Os métodos de realocação iterativa começam com uma partição inicial P_m^0 do conjunto E e a cada iteração geram novas partições P_m^k tais que

$$f(P_m^k) \leq f(P_m^{k-1})$$

onde f é uma função que traduz matematicamente a semelhança dentro de cada grupo das partições formadas em cada iteração.

O processo é interrompido quando algum teste de convergência é satisfeito, indicando que uma melhoria da partição não poderia mais ser obtida. O algoritmo é, também, de natureza heurística, constituindo sua solução em um ótimo local, que não corresponde, necessariamente, em um ótimo global. Uma questão importante é quanto a forma de geração da solução inicial para o algoritmo. Lucas [27] cita quatro maneiras de se efetuar a escolha da solução inicial:

- por escolha intencional;
- por seleção aleatória

- através de um método hierarquizado
- através de pontos semente

Abordaremos o critério de escolha intencional que, como veremos adiante, será utilizado no algoritmo para o PRA. Andeberg [2] cita que a escolha intencional pode ser feita com base em uma determinada variável que o analista considere como sendo mais importante.

Um outro fator que distingue os diferentes métodos de realocação iterativa é quanto ao número de grupos formados. Alguns métodos trabalham com um número fixo de grupos, conhecido a priori, outros trabalham de maneira a produzir um número de grupos final que não pode ser previsto com antecendência. Estaremos interessados naqueles métodos que trabalham com um número fixo de grupos e, em especial, no algoritmo de Forgy [16] que passamos a apresentar:

Método de Forgy

passo 0: Faça $k = 0$

passo 1: Se $k = 0$, tome uma partição P_m^0 de E;
senão, forme uma nova partição P_m^k de E, alocando cada elemento ao grupo cujo ponto semente esteja mais próximo e faça $k = k + 1$

passo 2: Calcule as centróides desses grupos, que serão os pontos sementes para os novos grupos

passo 3: Se $f(P_m^k) = f(P_m^{k-1})$, pare;
senão, vá ao passo 1

A solução inicial é fornecida para o algoritmo juntamente com os centróides dos m grupos, sendo estes, os pontos sementes para a geração de novos grupos. A função objetivo utilizada, $f(P_m^k)$, é a soma dos quadrados dentro dos grupos, definida anteriormente, objetivando minimizar as distâncias dos elementos às centróides de seus grupos. As sequencias de passos 1 e 2 geram novas partições tais que

$$f(P_m^k) \leq f(P_m^{k-1})$$

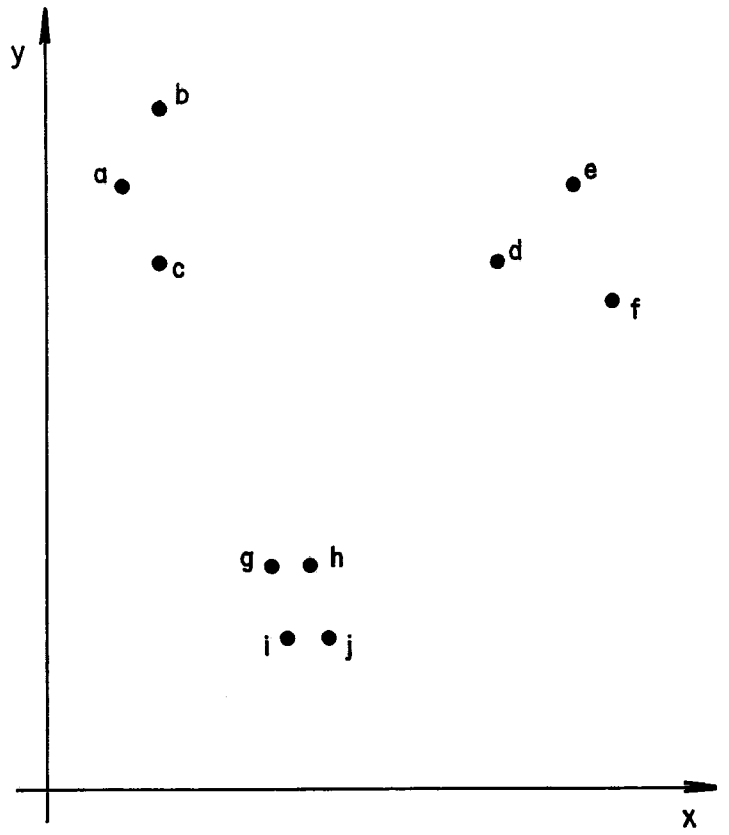
A convergência do método de Forgy, segundo Lucas [27], é garantida. Uma particularidade interessante deste método é a rapidez com que o algoritmo converge para a solução ótima local. Consideremos a figura IV.1 representando os pontos do conjunto E = {a,b,...,j} situados no R^2. De posse das coordenadas retangulares dos elementos do conjunto E, vamos aplicar o método de realocação iterativa para formamos três grupamentos distintos a partir de E:

inicialização:
k = 0
Tomemos a patição inicial

$$P_3^0 = \{ \underset{g_1^0}{\{a,g\}}, \underset{g_2^0}{\{b,e,h,i\}}, \underset{g_3^0}{\{c,d,f,j\}} \}$$

centróide dos grupos de P_3^0 :

$$XC_i = \frac{\sum_{j \in g_i} X_j}{n_i} \qquad YC_i = \frac{\sum_{j \in g_i} Y_j}{n_i}$$



PARTIÇÃO INICIAL

$g_1 = \{a , g\}$

$g_2 = \{b , e , h , i\}$

$g_3 = \{c , d , f , j\}$

a - (10, 80)	d - (60, 70)	g - (30, 30)	j - (37, 20)
b - (15, 90)	e - (70, 80)	h - (35, 30)	
c - (15, 70)	f - (75, 65)	i - (32, 20)	

FIG. IV. 1 - APLICAÇÃO DO MÉTODO DE FORGY.

$XC_1 = 20,0000 \qquad YC_1 = 55,0000$
$XC_2 = 38,0000 \qquad YC_2 = 55,0000$
$XC_3 = 46,7500 \qquad YC_3 = 56,2500$

Valor da função objetivo:

$$f(P_3^0) = \sum_{i=1}^{3} W_i = \sum_{i=1}^{3} \sum_{j \in g_i} [(X_j - XC_i)^2 + (Y_j - YC_i)^2]$$

$f(P_3^0) = 10.593,5000$

Realocando cada nó para o grupo cujo centróide está mais próximo teremos:

$k = 1$

obtemos uma nova partição para o conjunto E,

$$P_3^1 = \{ \underset{g_1^1}{\{a,b,c\}} \quad \underset{g_2^1}{\{g,h,i,j\}} \quad \underset{g_3^1}{\{d,e,f\}} \}$$

centróide dos grupos de P_3^1 :

$XC_1 = 13.3333 \qquad YC_1 = 80.0000$
$XC_2 = 33.5000 \qquad YC_2 = 25.0000$
$XC_3 = 68.3333 \qquad YC_3 = 71.6667$

valor da função objetivo para P_3^1 :

$f(P_3^1) = 579$

como $f(P_3^1) < f(P_3^0)$ continuamos o processo iterativo, fazendo

$k = 2$

de forma semelhante ao passo anterior, realocamos

os nós aos grupos de centróides mais próximos e teremos:

$P_3^2 = P_3^1$, ou seja não houve, efetivamente, nenhuma mudança em relação à partição anterior e, por conseguinte temos que

$$f(P_3^2) = f(P_3^1) = 579$$

e o algoritmo convergiu para a solução P_3^2 que é um ótimo local para o problema. O gráfico da figura IV.2 apresenta a solução final encontrada pelo algoritmo.

O exemplo anterior mostra a facilidade das operações envolvidas no algoritmo. Não se pode garantir o número de iterações necessárias para a convergência do algoritmo, mas nota-se pelo exemplo, a rapidez com que o algoritmo chegou ao ótimo local e a brusca redução da função objetivo já na primeira iteração. Em cada iteração, a tentativa de realocação dos n elementos do conjunto E faz com que o algoritmo tenha que efetuar o cálculo de n.m distâncias e n.(m-1) comparações de distâncias. Uma vez que o número de iterações até a convergência e nem mesmo um limite superior são conhecidos, não podemos elaborar uma análise da complexidade de pior caso para o algoritmo de Forgy. Face a este problema, programamos o algoritmo, programa REALOCIT, apresentando na tabela IV.3 resultados empíricos que mostram a performance do referido algoritmo. Geramos, aleatóriamente, a partir de suas coordenadas retangulares, X e Y, vinte problemas com M = 5 (número de grupos) e N = 50 (total de nós). As coordenadas, X e Y, foram geradas nos intervalos [10,610] e [10,190], respectivamente, segundo uma distribuição uniforme de probabilidades de se tomar qualquer um dos valores, nos respectivos intervalos. Listamos na tabela IV.3 o número de

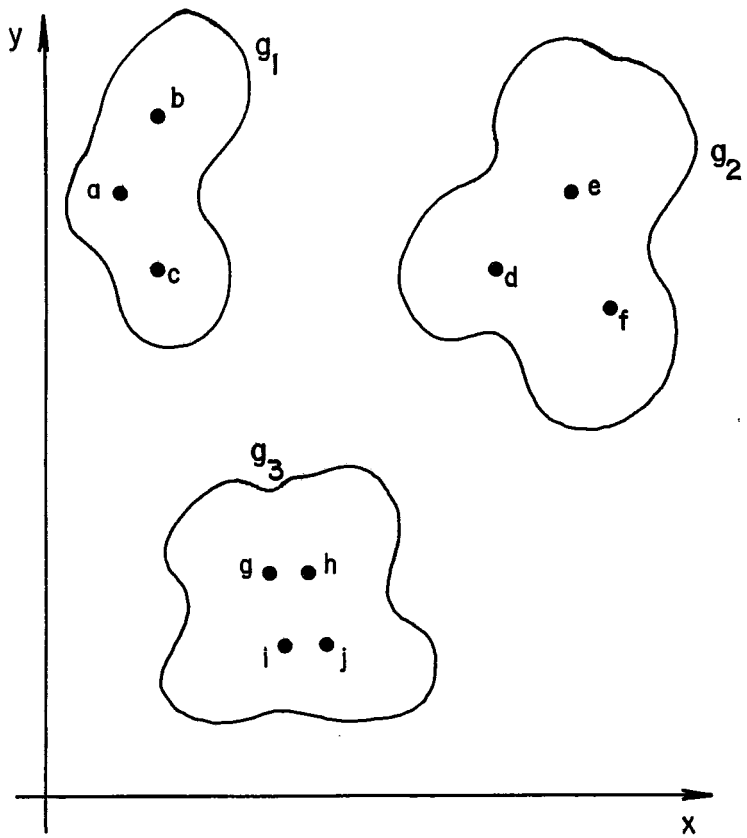


FIG. IV.2 - RESULTADO FINAL DA APLICAÇÃO DO ALGORITMO DE FORGY.

Num. (1)	$f(P_5^0)$ (2)	$f(P_5^*)$ (3)	Iter. (4)	Tempo (5)
1	708.691	119.178	6	4
2	820.040	90.149	6	4
3	731.779	147.860	6	4
4	841.714	125.298	8	5
5	749.336	97.296	6	4
6	685.056	103.083	8	5
7	763.808	113.565	5	3
8	657.776	116.193	5	3
9	770.427	125.010	9	6
10	665.449	113.139	5	3
11	770.459	128.192	4	2
12	689.137	130.014	5	3
13	611.325	99.600	8	5
14	706.974	162.385	4	2
15	735.652	166.149	3	2
16	768.678	107.811	8	5
17	675.567	88.082	8	5
18	791.563	128.522	6	4
19	667.738	121.215	6	4
20	577.209	113.045	5	3

Tabela IV.3- resultados computacionais para o método de realocação iterativa

(1)- número do problema

(2)- valor da função objetivo para a partição inicial

(3)- valor da função objetivo para a partição final, ótimo local encontrado pelo método

(4)- número de iterações do algoritmo

(5)- tempo de processamento em segundos para um microcomputador PC-XT

iterações necessárias à convergência, os valores iniciais e finais da função objetivo (soma dos quadrados dentro dos grupos) e o tempo de processamento em um microcomputador PC-XT. As partições iniciais foram geradas, também, aleatoriamente. Pode-se ver na referida tabela que o algoritmo necessitou, em média, de seis iterações para chegar ao ótimo local e demandou o tempo médio de três segundos para os problemas.

Na seção que se segue, quando tratarmos do nosso algoritmo para a resolução do Problema dos Auditores, utilizaremos a mesma idéia dos métodos de realocação iterativa para formar grupamentos de agências em torno das diversas praças-sedes dos auditores.

IV.3.2 - Um algoritmo para o PRA - SISROTAU

Desenvolvemos um algoritmo heurístico original para resolver o Problema de Roteamento de Auditores, doravante denominado SISROTAU. Nosso algoritmo utiliza um método de realocação iterativa para agrupar as agências em torno das diferentes praças-sedes e um procedimento sequencial do tipo saving para sequenciar as agências já agrupadas. A forma como combinamos as etapas de AGRUPAR E SEQUENCIAR distingue o nosso algoritmo daqueles encontrados na literatura. No capítulo III, quando tratamos do PRVM citamos algoritmos que utilizam a estratégia de promover um agrupamento dos nós em torno dos depósitos e, posteriormente, sequenciam os nós dentro dos grupos formados, dividindo, portanto, o problema em problemas do tipo PRV1. Tal estratégia não nos pareceu eficiente para o PRA, onde o número de praças-sedes (equivalentes depósitos) é bastante elevado, cerca de duzentas praças-sedes distintas, onde cada praça-sede dará origem a uma pequena quantidade de rotas, em geral, uma rota. De forma diferente da estratégia exposta acima, o SISROTAU promove "agrupamentos" e "sequenciamentos" a cada iteração. Tal estratégia se mostra eficiente para o PRA. Nas páginas seguintes apresentamos o SISROTAU em detalhes.

Na modelagem definimos um problema com M praças-sedes, NA auditores, NAS agências do tipo missão simples e NAC agências do tipo missão conjunta. Fazemos, agora, uma pequena modificação em tal definição, tomando o número de praças-sedes como sendo igual ao número de auditores por duplicar, triplicar, etc. aquelas praças sedes que tenham mais de um auditor. Então, definiremos, agora, um problema com M auditores, dispensando NA da nossa terminologia e assumindo que os nós são assim representados:

A = {1,2,...,M} = conjunto de Auditores
S = (M+1,...,M+NAS} = conjunto das agências de missão simples
C = {M+NAS+1,...,n} = conjunto das agências de missão conjunta

O fato de estarmos designando um nó para cada auditor não produz um aumento significativo do número de nós que teremos de armazenar, uma vez que a maioria dos auditores do Banco do Brasil S.A. estão localizados em praças-sedes distintas. Não seria o caso dos Problemas de Roteamento de Veículos onde, em geral, temos um número pequeno de depósitos contendo vários veículos.

O funcionamento do algoritmo é baseado na aplicação de quatro procedimentos básicos : INICON, AGRUPA, SEQUENCIA, REFIN1 e REFIN2. Descreveremos isoladamente cada um dos procedimentos e, em seguida, o funcionamento conjunto dos mesmos, no que resultará o algoritmo.

PROCEDIMENTO INICON

O procedimento INICON trata de efetuar as ligações do tipo missão conjunta. Nos vimos, na seção IV.2, que para a ligação das missões conjuntas tinhamos que considerar a simultâneidade da chegada dos auditores nas agências daquele tipo. O procedimento INICON é o primeiro procedimento aplicado pelo algoritmo e efetua dois tipos de ligações:

a)- missão conjunta do tipo I: aquelas em que teremos uma única agência do tipo missão conjunta como sendo a primeiro nó das rotas de dois auditores não impedidos (veja figura IV.4.a).

b)- missão conjunta do tipo II: aquelas em que teremos duas agências do tipo missão conjunta como sendo as duas primeiras agências nas rotas de dois auditores não impedidos para ambas as agências (veja fig. IV.4.b).

Temos, obviamente, a condição de simultâneidade de chegada satisfeita para os dois tipos de missões conjuntas formadas por INICON. Embora a modelagem do PRA não limite as missões conjuntas aos tipos I e II, o algoritmo limitará no sentido de evitar que o PRA ganhe fortes características de 'scheduling', tornando-o, ainda, mais difícil. Colocando o procedimento em forma algoritmica teremos:

passo 1: Para todos os pares de agências do tipo missão conjunta, AG1 e AG2 $\in$ C, ainda não roteadas e tais que d(AG1,AG2) $\leq$ RMAX, tome, se existirem, os dois auditores, ainda não roteados e não impedidos para as agências AG1 e AG2, AU1 e AU2 que produzem os dois menores valores W1(AG1,AG2,AU)= d(AU,AG1) + d(AG1,AG2) + d(AG2,AU). Tome, se existirem, as agências $AG1^*$ e $AG2^*$ e os correspondentes auditores, AU1 e AU2, tal que W(AG1,AG2) = W1(AG1,AG2,AU1) + W1(AG1,AG2,AU2) seja mínimo e W(AG1,AG2) $\leq$ DISTMAX. Se existir tal mínimo vá para o passo 2. Caso contrário, vá ao passo 3.

passo 2: Promova as ligações correspondentes às duas missões conjuntas do tipo II recém formadas, ou seja:
Rota 1 = AU1 - $AG1^*$ - $AG2^*$ - AU1
Rota 2 = AU2 - $AG1^*$ - $AG2^*$ - AU2
volte ao passo 1.

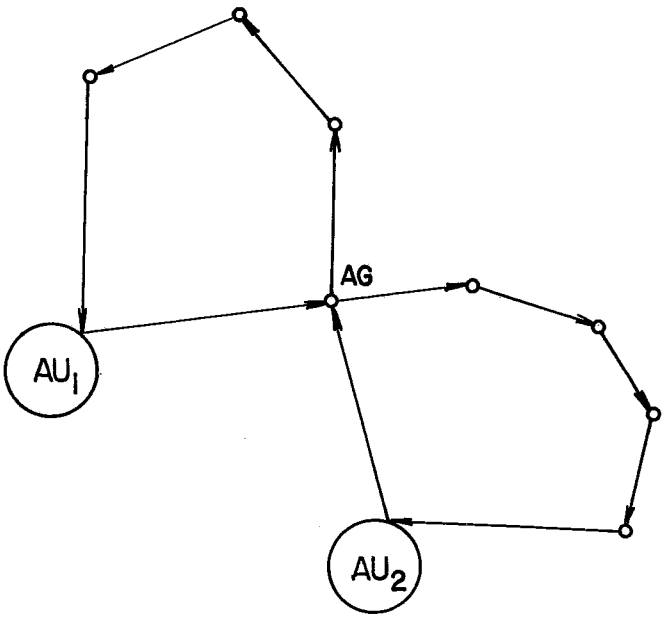


FIG. IV. 4a - MISSÃO CONJUNTA TIPO I.

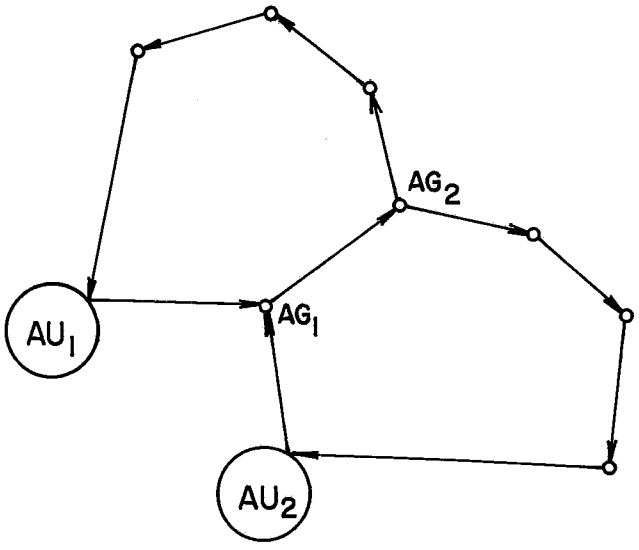


FIG. IV. 4a - MISSÃO CONJUNTA TIPO II.

passo 3: Comece a formar missões conjuntas do tipo I. Para toda agência AG $\in$ C, ainda não roteada, tome os auditores AU1 e AU2 mais próximos e não impedidos para AG. Tome a agência AG^* que maximiza a expressão W2 (AG) = 2*d(AU1,AG) + 2*d(AU2,AG). Vá ao passo 4.

passo 4: Efetue as ligações correspondentes às duas rotas de missão conjunta tipo I, ou seja:
Rota 1 = AU1 - AG^* - AU1
Rota 2 = AU2 - AG^* - AU2
Se houver agências do tipo missão conjunta por rotear volte ao passo 3. Caso contrário, termine o procedimento INICON.

RMAX e DISTMAX são parâmetros fornecidos a priori para o algoritmo, estando, ambos, relacionados com os passos 1 e 2 do procedimento INICON, onde se tentará formar missões conjuntas do tipo II. RMAX restringe a busca pelos pares de agências AG1 e AG2 no sentido de economizar tempo de processamento. DISTMAX limita o tamanho de duas novas rotas do tipo missão conjunta que serão formadas. Na verdade, tais rotas foram inicializadas nesta fase, podendo, posteriormente, serem aumentadas pelos demais procedimentos do algoritmo. Desta forma, deve-se tomar um valor de DISTMAX compátível com tal situação.

O algoritmo dá preferência para a inicialização das missões conjuntas do tipo II e efetiva tantas quantas os parâmetros RMAX e DISTMAX permitirem. As demais agências de missão conjunta são colocadas em rotas do tipo I. É importante lembrar que estamos supondo que haja auditores suficientes para a formação de tais missões.

Mais uma vez salientamos que o procedimento

inicializa algumas rotas que ainda não são definitivas, pois poderão ser expandidas pelos demais procedimentos do algoritmo.

PROCEDIMENTO AGRUPA

O procedimento AGRUPA nada mais é do que uma pequena variação do método de Forgy apresentado anteriormente. AGRUPA tratará dos nós que representam as agências de missão simples. AGRUPA atuará, a cada iteração do algoritmo, sobre um número decrescente de grupos, minimizando a soma dos quadrados dentro dos mesmos. O número de grupos decresce na medida em que alguns deles são sequenciados e, portanto, não mais considerados pelo procedimento. Os grupos a que referimos são constituídos de uma praça-sede (ou auditor) e agências do tipo missão simples ou conjunta. Mais a frente, veremos que a partição inicial de todo o conjunto de auditores e agências é obtida nos dois primeiros passos do SISROTAU, tomando-se as praças-sedes como pontos sementes dos M grupos.

De maneira diferente do método de Forgy, nós permitimos, tão somente, as realocações dos nós que correspondem a agências de missão simples e consideramos, ainda, as restrições de impedimento para evitar realocações que não sejam viáveis.

Procedimento SEQUENCIA

O procedimento SEQUENCIA, todas as vezes que for solicitado, atuará sobre os nós de um grupamento para sequenciar a rota do auditor

pertencente ao mesmo. Tomamos para o procedimento sequência uma pequena variante do algoritmo de Mole & Jameson apresentado no capítulo III, quando descreviamos o Problema de Roteamento de Veículos - caso 1-depósito. Tal variante apresenta duas diferenças em relação ao algoritmo original de Mole & Jameson:

i)- A inicialização de cada rota é feita da seguinte forma:

a)-para as rotas que são do tipo missão conjunta, inicializadas por INICON, a inicialização é feita seguindo o critério de menor esforço de inclusão, preservando, contudo, as posições das agências de missão conjunta na posição originalmente designada por INICON.

b)-para as rotas do tipo missão simples e, portanto, ainda não inicializadas, colocamos o nó correspondente à agência de missão simples mais próxima ao auditor e não impedida para o mesmo.

ii) - Não aplicamos a rotina 2-opt após cada inclusão de nó sobre a rota, mas sim quando a rota está terminada em virtude da capacidade máxima de tempo do auditor que lhe dá origem. A razão desta modificação se deve ao fato de que, para o problema dos auditores, o número de nós por rotas é pequeno e, então, a experiência mostra que os resultados são práticamente iguais. Com isso economizamos tempo de processamento. Ainda com relação à rotina 2-otm, salientamos que modificamo-la para que possa atuar sobre rotas do tipo missão conjunta preservando a primeira (missão tipo I) ou as primeiras (missão tipo II) posições das rotas de missão conjunta.

É importante ressaltar que o procedimento SEQUENCIA considera as restrições de impedimentos

antes de rotear as agências dentro dos grupos formados por AGRUPA. Mais adiante, veremos que existe a possibilidade de que uma ou outra agência possa entrar em um grupamento para o qual esteja impedida.

Procedimento REFIN1

O procedimento REFIN1 entrará em ação uma vez que todos os auditores tenham sido sequenciados e algumas poucas agências de missão simples tenham sido ignoradas no processo. O procedimento pode ser representado por:

```
Para AG:=M+1 to M+NAS faça
Se (AG não foi roteada) faça
início
    procure o auditor Au, não impedido, com
    capacidade de tempo suficiente para receber AG
    e mais próximo da agência.
    Se existir AU faça
    início
        coloque a agência AG sobre a rota de AU,
        ligada na última posição da rota (entre a
        última agência sobre a rota e a
        praça-sede, chamando em seguida o
        procedimento 2-otm.
    fim
    senão
    deixe a agência no grupo das agências que serão
    roteadas manualmente
fim
```

A representação algoritmica de REFIN1 nos revela a possibilidade de que alguma agência pode chegar ao final do algoritmo sem ter sido roteada. Isto tende a acontecer na medida em que se tem um número muito grande de impedimentos para todos os

auditores. Uma vez acontecida tal situação, deixamos as agências para serem roteadas por procedimento manual/visual, ou seja, o operador do sistema promoverá a inclusão de tais agências. Ressaltamos, entretanto, que tal situação dificilmente aconteceria nos problemas práticos a que o SISROTAU se destina.

Procedimento REFIN2

Este é o nosso último procedimento e, também, o último procedimento a ser aplicado sobre os nós da rede na sequência do algoritmo. REFIN2 tentará melhorar as rotas pela troca, sucessiva, de dois nós, do tipo missão simples, ligados sobre rotas distintas.

Em linguagem algoritmica teremos:

```
Trocou:= verdade
Enquanto (Trocou = verdade) e (Tempo < TMAXREFIN)
faça
início
    trocou:=falso;
    Para AG1:= M+1 to M+NAS faça
    Para AG2:= AG1+1 to M+NAS faça
    Se a troca das posições de AG1 e AG2 não
    inviabiliza os tempos das duas rotas
    envolvidas, as restrições de impedimentos não
    são violadas e a troca permite uma economia na
    distância total de roteamento faça
    início
        troca:=verdade
        troque as posições de AG1 e AG2
    fim
fim
```

Notamos que o fim do procedimento REFIN2 será determinado por qualquer uma das duas condições abaixo:

i)- Não se consegue mais trocar, vantajosamente, nenhum par de agências

ii)-O tempo máximo permitido, TMAXREFIN, esgotou-se.

Uma vez que já apresentamos os cinco procedimentos básicos do algoritmo, passamos a descrever o seu **funcionamento global:**

passo 0: Alocamos todas as agências de missão simples para os auditores mais próximos às mesmas e não impedidos, sem nos preocuparmos com o TMAXROT e vamos ao passo 1.

passo 1: Chamamos o procedimento INICON para inicializar as missões conjuntas do tipo I ou II. Ao final destes dois primeiros passos teremos os grupamentos G_i (i=1,..,M) associados às respectivas praças-sedes. Alguns destes podem conter uma rota já inicializada por INICON, outros conterão tão somente agências do tipo missão simples. Aplicamos o procedimento AGRUPA e vamos ao passo 2.

passo 2: Se todas as agências do tipo missão simples foram roteadas, vá ao passo 5. Caso contrário, se em todos os grupos ainda não sequenciados, não existe excesso de tempo alocado (em relação a TMAXROT) vamos ao passo 4. Caso contrário, passo 3.

passo 3: Para todo grupo G_i, ainda não sequenciado, cuja soma dos tempos das agências a ele alocado é igual ou superior a TMAXROT, aplique o procedimento SEQUENCIA. Se existir algum grupo (auditor), K , ainda não sequenciado, aloque ao mesmo, todas as agências que, porventura, tenham ficado em excesso nos grupos recém-sequenciados. Aplique o procedimento AGRUPA sobre os grupos remanescentes (ainda não sequenciados) e volte ao passo 2.
Se não existir o referido grupo K, vá ao passo 5.

passo 4: Aplique o procedimento SEQUENCIA para sequenciar todos os grupos, ainda não sequenciados, e vá ao passo 5.

passo 5: Aplique REFIN1 para tentar alocar as agências que, porventura, não tenham sido roteadas e vá ao passo 6.

passo 6: Aplique REFIN2 para tentar refinar a solução obtida pelos passos anteriores e termine o processo.

Na descrição do procedimento SEQUENCIA destacamos a possibilidade de existência de agência em grupo para o qual esteja impedida. Tal fato acontece em virtude de, no passo 3, quando procuramos pelo grupo K, não testarmos as restrições de impedimento. Fazemos desta forma para conseguir maior rapidez na realocação daquelas agências que ficaram em excesso nos grupamentos recém-sequenciados no mesmo passo. Em contrapartida, poderíamos, evidentemente, chegar a uma situação final em que o último grupo, ainda não

sequenciado, ter uma ou outra agência para as quais existem impedimentos. Uma vez chamado o procedimento SEQUENCIA, tais agências não seriam consideradas no sequenciamento, restando, entretanto, a possibilidade de serem sequenciadas por REFIN1. Se este último procedimento não conseguir, deixamos tais agências para serem roteadas com a iteração do operador do sistema de roteamento.

De modo a deixar claro o funcionamento do SISROTAU, apresentaremos um exemplo numérico em que temos que efetuar o roteamento de quatro auditores, dezoito agências do tipo missão simples e uma agência do tipo missão conjunta, representados na figura IV.5. Os nós {1,2,3,4} representam as praças-sedes, os nós {5,6,...,22} as agências de missão simples e o nó 23 representa a agência de missão conjunta. Todas as agências demandam um tempo igual a 14 dias para serem roteadas e as suas coordenadas podem ser vistas na própria figura IV.5. Consideramos TMAXROT = 70, LAMBDA = 1.5, MI = 1 e TMAXREFIN = 1 min. Os demais parâmetros, RMAX, DISTMAX não terão nenhuma influência na resolução do nosso exemplo, uma vez que estes estão relacionados com a construção de missão conjunta do tipo II, não existente no nosso exemplo. Supomos, ainda, que não existe impedimento para os auditores. Passemos, agora, à resolução:

inicialização: passo 0 e passo 1

O passo 0 designa cada agência de missão simples para o seu auditor mais próximo e teremos:

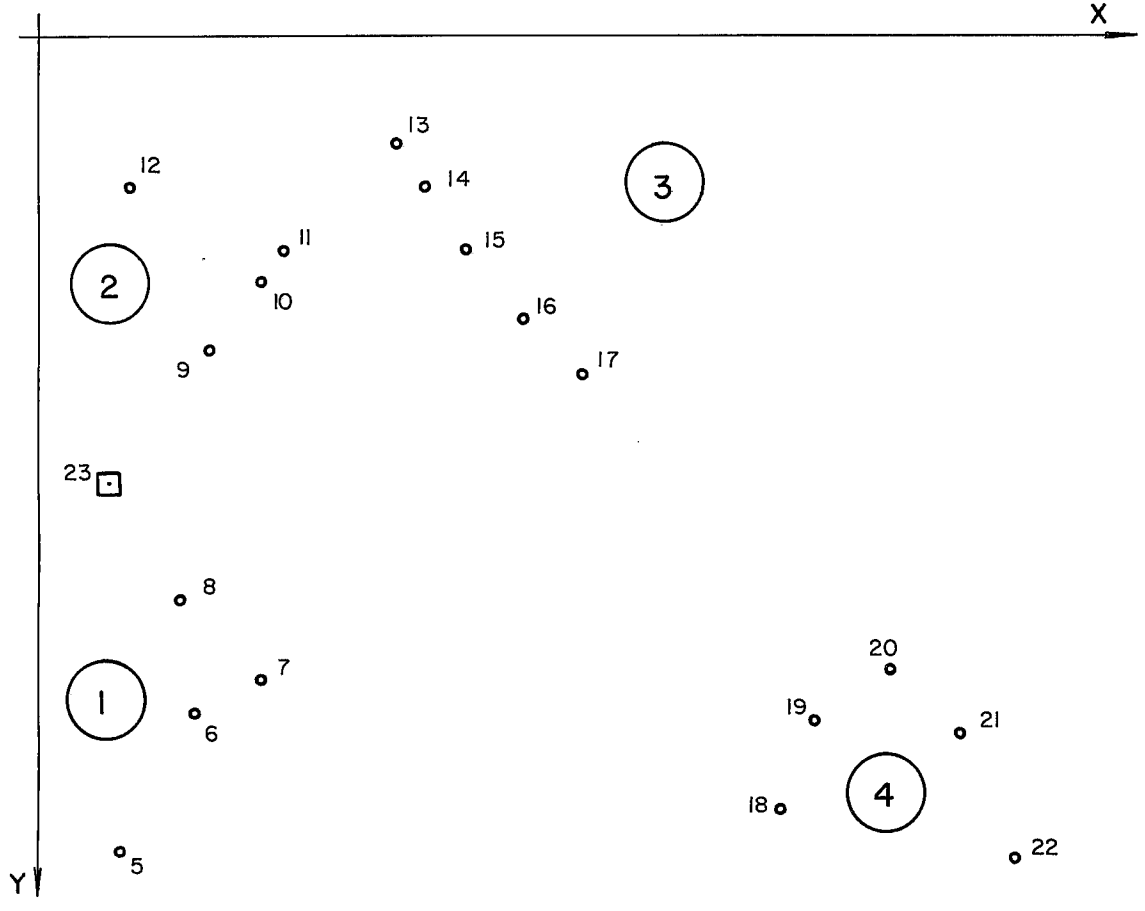
G_1 = {5,6,7,8}

G_2 = {9,10,11,12,13,14,15,16}

G_3 = {17}

G_4 = {18,19,20,21,22}

onde G_1, G_2, G_3 e G_4 são os grupamentos formados em torno dos auditores 1,2,3 e 4 respectivamente.



COORDENADAS DOS NÓS

1- (50 , 150)	13- (150 , 38)
2- (50 , 50)	14- (155 , 45)
3- (450, 50)	15- (169 , 56)
4- (450, 150)	16- (200 , 67)
5- (55 , 155)	17- (270 , 76)
6- (67 , 132)	18- (421 , 148)
7- (78 , 126)	19- (432, 134)
8- (65 , 113)	20- (461 , 125)
9- (69 , 72)	21- (472, 135)
10- (78 , 61)	22- (481 , 156)
11- (82 , 56)	23- (54 , 94)
12- (56 , 45)	

FIG. IV. 5 - EXEMPLO PARA O SISROTAU.

O passo 1 promove o roteamento da agência 23, efetuando uma missão conjunta do tipo I com os auditores 1 e 2, ou seja:

1 → 23 → 1
2 → 23 → 1

após a aplicação do procedimento AGRUPA teremos:
G_1 = {5,6,7,8,23}
G_2 = {9,10,11,12,13,14,15,16,23}
G_3 = {17}
G_4 = {18,19,20,21,22}

lembramos que nesta fase os grupos 1 e 2, embora não estejam sequenciados definitivamente, já foram inicializados com missão conjunta.

Entramos, agora, no processo iterativo, passando ao passo 2. Nesta etapa verificamos que nenhuma agência do tipo missão simples foi sequenciada e que os grupos G_1, G_2 e G_4 tem um tempo alocado maior ou igual a TMAXROT e, portanto, serão sequenciados nesta etapa.

Aplicando o procedimento SEQUENCIA para os grupos G_1, G_2 e G_4 teremos as seguintes rotas formadas:

a) 1 → 23 → 8 → 7 → 6 → 5 → 1
   tempo da rota = 70
   distancia = 142

b) 2 → 23 → 9 → 10 → 11 → 13 → 2
   tempo da rota = 70
   distancia = 262

c) 4 → 22 → 21 → 20 → 19 → 18 → 4
   tempo da rota = 70
   distancia = 147

Em seguida verificamos que as agências 12, 14, 15 e 16 ficaram em excesso nos três grupos recém-sequenciados. Realocamos estas quatro agências para o grupo G_3 ainda não sequenciado. Aplicamos o procedimento AGRUPA e teremos:
G_3 = {12,14,15,16,17}
os demais grupos já foram sequenciados e não são mais considerados. Voltando ao passo 2, verificamos que não existe excesso de tempo alocado no único grupo ainda não sequenciado e passamos ao passo 4, aplicando o procedimento SEQUENCIA para G_3:

3 → 14 → 12 → 15 → 16 → 17 → 3
tempo da rota = 70
distancia = 794

Uma vez que todas as agências foram sequenciadas o passo 5 não tem qualquer função e passamos, então, ao passo 6. O passo 6 efetua, sucessivamente as seguintes trocas antes de convergir:
primeira: nó 10 (rota 2) <==> nó 12 (rota 3)
segunda : nó 10 (rota 3) <==> nó 13 (rota 2)

O algoritmo termina com as seguintes rotas:

1 → 23 → 8 → 7 → 6 → 5 → 1
tempo de rota = 70
distância = 142
2 → 23 → 9 → 10 → 11 → 12 → 2
tempo de rota = 70
distância = 127
3 → 13 → 14 → 15 → 16 → 17 → 3
tempo de rota = 70
distância = 613
4 → 22 → 21 → 20 → 19 → 18 → 4
tempo de rota = 70
distância = 147

que podem ser visualizadas na figura IV.6.

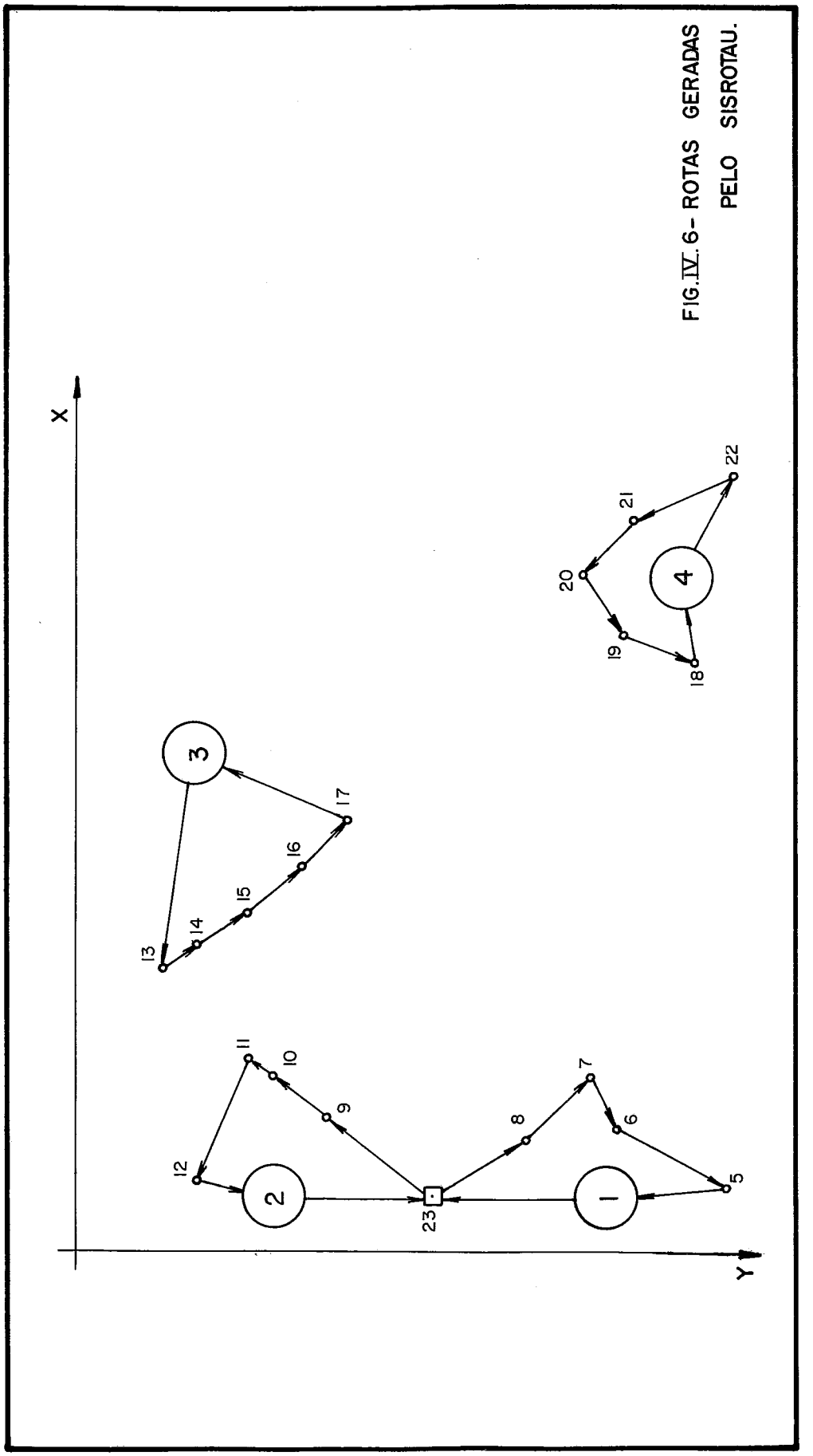


FIG. IV. 6- ROTAS GERADAS PELO SISROTAU.

Programamos o algoritmo acima descrito em Turbo Pascal 4.0, programa SISROTAU, utilizando as coordenadas retangulares X e Y dos nós para o cálculo das distâncias entre os diversos pares de nós. A entrada de dados para o programa, constitui-se, portanto, das coordenadas retangulares dos nós, dos tempos t_i de auditoria para cada agência e da matriz de impedimentos para os auditores. Além dos dados da rede, o programa exige os seguintes parâmetros:

- LAMBDA e MI = parâmetros relacionados com o procedimento SEQUENCIA e definidos no capítulo III, quando apresentamos o algoritmo de Mole & Jameson.

- TMAXROT = parâmetro que limita o tempo de todas as rotas

- TMAXREFIN = parâmetro que limita o tempo de funcionamento do procedimento REFIN2

- RMAX e DISTMAX = parâmetros relacionados com o procedimento INICON para formação de missões conjuntas do tipo II.

O programa permite, ainda, que sejam variados os parâmetros para, com os dados de uma mesma rede, fornecer soluções diferentes e, evidentemente, dando a oportunidade ao analista de tomar a melhor delas. O SISROTAU apresenta, ainda, uma rotina gráfica para que o analista possa visualizar na tela do vídeo o conjunto de rotas formadas ao final de uma rodada do programa.

IV.4. Resultados Computacionais para o SISROTAU

Nesta seção apresentamos uma série de resultados computacionais para o SISROTAU de maneira a avaliar empiricamente a sua performance. O primeiro teste trata de problemas de roteamento de veículos 1-depósito retirados da literatura. Aqui, evidentemente, estamos considerando a analogia existente entre o PRV1, apresentado no capítulo III, com o PRA. Neste primeiro teste não consideramos o procedimento INICON e as restrições de impedimento constantes do SISROTAU de modo a compatibilizar totalmente o PRV1 com o PRA. Os problemas, originalmente problemas de roteamento de veículos 1-depósito, podem ser entendidos como problemas de auditores localizados em uma mesma praça-sede. O quadro IV.7 apresenta as dimensões de nove problemas retirados da literatura citando as fontes.

Sob as colunas A, B, C e D do quadro IV.7 vemos os resultados fornecidos pelos algoritmos de Clarke Wright (saving), Mole & Jameson, o algoritmo sweep de Gillett & Miller e SISROTAU, respectivamente, apresentando o número de rotas geradas entre parênteses.

O quadro IV.8 apresenta, de maneira mais detalhada, o desempenho do SISROTAU para os problemas do quadro IV.7. Sob a coluna DISTANT o valor da solução (distância total de roteamento) antes da aplicação do procedimento REFIN2, sob DISTFINAL a solução final do problema e o número de rotas entre parênteses, sob TREFIN o tempo em segundos de processamento do procedimento REFIN2 e sob TTOTAL o tempo total de processamento de SISROTAU também em segundos. O processamento foi feito em um microcomputador PC-XT. A análise dos quadros IV.7 e IV.8 nos revela a proximidade entre as soluções geradas pelo SISROTAU e os demais

Num.	Num.nós	A	B	C	D
1	120	1079(7)	1100(7)	1266(7)	1088(7)
2	100	831(10)	879(10)	937(10)	833(10)
3	50	585(6)	-	546(5)	578(6)
4	75	900(10)	-	865(10)	884(11)
5	100	887(8)	-	862(8)	893(8)
6	75	-	-	1127(15)	1093(15)
7	75	-	-	754(8)	803(8)
8	75	-	-	715(7)	752(7)
9	75	-	-	1170(14)	1172(14)

fonte dos problemas:

1 e 2 ==> problemas 11 e 12 de Christofides [12]

3, 4 e 5 ==> problemas 8, 9 e 10 de Christofides [10]

6,7,8 e 9 ==> problemas 6,8,9 e 10 de Gillett [18].

A - método saving

B - Mole & Jameson

C - sweep (Gillett and Miller)

D - SISROTAU

Quadro IV.7 - problemas da literatura

NUM	DISTANT	DISTFINAL	TREFIN	TTOTAL
1	1088	1088(7)	5	108
2	855	833(10)	20	88
3	578	578(6)	2	35
4	906	884(11)	11	71
5	907	893(8)	13	107
6	1103	1093(15)	9	81
7	819	803(8)	9	71
8	767	752(7)	10	60
9	1203	1172(14)	16	147

Quadro IV.8 - Resultados para o SISROTAU

algoritmos comparados.

Em seguida geramos, aleatóriamente, vinte problemas do tipo PRV1, resolvendo-os pelos algoritmos Saving , Mole & Jameson e SISROTAU. Os problemas possuem 70 (setenta) nós (agências ou consumidores) e um depósito central (ou praça-sede) contendo dez veículos disponíveis (ou auditores) e o TMAXROT é igual a 190. A coordenada X de cada nó é gerada aleatóriamente no intervalo [10,600] e a coordenada Y, no intervalo [10,200]. Os tempos t_i (ou carga no caso de PRV1) são gerados no intervalo [20,30]. As coordenadas do depósito central são fixadas no ponto médio dos intervalos citados anteriormente e o processo de geração aleatória supõe uma distribuição uniforme das probabilidades de se tomar qualquer um dos valores dentro dos respectivos intervalos. Com relação à geração aleatória de problemas do tipo PRV1 é importante dizer que nós a fazemos de maneira a obter problemas semelhantes aos problemas apresentados pelos autores citados no quadro IV.9.

No quadro IV.9 apresentamos os resultados obtidos no processamento dos vinte problemas em um microcomputador PC-XT. Sob as colunas SAVING, MOLE&JAMESON e SISROTAU apresentamos as distâncias finais obtidas pelos algoritmos e os tempos de processamento, em segundos, entre parênteses. Utilizemos o 'teste de hipótese para médias', detalhado no ANEXO I, para compararmos o SISROTAU com os outros dois algoritmos:

a) SISROTAU X MOLE&JAMESON

Tomemos o nível α = 1 % de significância para realizar o teste. Seja d_i a diferença entre as soluções obtidas, para cada problema, entre a coluna MOLE&JAMESON e SISROTAU, nesta ordem. Uma vez que não conhecemos o desvio padrão da

NUM.	SAVING	MOLE&JAMESON	SISROTAU
1	4804(10)	4905(8)	4800(30)
2	5035(10)	5159(7)	5036(29)
3	4453(10)	4511(7)	4447(29)
4	4567(10)	4574(8)	4651(27)
5	4453(10)	4613(7)	4531(35)
6	4385(10)	4577(7)	4445(30)
7	4740(10)	5453(7)	5034(35)
8	4889(10)	5235(7)	5006(30)
9	4475(10)	4503(7)	4415(32)
10	4418(9)	4733(7)	4467(27)
11	4837(10)	4883(7)	4688(37)
12	4982(10)	5230(7)	5018(30)
13	4487(10)	4531(7)	4567(30)
14	4624(10)	4853(7)	4525(26)
15	4544(10)	4841(7)	4605(33)
16	4387(10)	4584(7)	4529(31)
17	4637(10)	4770(7)	4760(34)
18	4436(10)	4615(7)	4656(30)
19	4658(10)	4954(7)	4570(32)
20	4459(10)	4571(7)	4494(32)

Quadro IV.9 - Problemas tipo PRV1

distribuição das médias destas diferenças, utilizamos a distribuição t de Student com o desvio da nossa própria amostra de diferenças d_i. Então, formulamos as duas hipótese básicas:

$H_0 : \bar{d} = 0$ (hipótese nula)

$H_1 : \bar{d} > 0$ (hipótese alternativa)

onde $\bar{d}$ é a média das diferenças d_i.
A hipótese nula sugere que os algoritmos em teste têm a mesma performance e a hipótese alternativa sugere que o algoritmo SISROTAU produz soluções

melhores do que o de MOLE&JAMESON. Passemos, portanto, ao cálculo:

$$\sum_{i=1}^{20} d_i = 3151 \qquad \text{e} \qquad \bar{d} = 157,5500$$

$$\sum_{i=1}^{20} d_i^2 = 816.365,0000$$

Calculamos, agora, o desvio padrão da amostra:

$$s_d^2 = \frac{816.365,0000 - \frac{(3151)^2}{20}}{19}$$

$$s_d = 129,7619$$

De posse de s_d calculamos a estatística t a ser testada com $t_{19,1\%}$ (valor crítico da distribuição de Student, com 19 graus de liberdade e $\alpha = 0.01$) que é igual a 2.5390. Calculemos t :

$$t = \frac{\bar{d}}{s_d / \sqrt{n}} = \frac{157,5500}{129,7619 / \sqrt{20}}$$

t = 5,4298

Verificamos, portanto, que $t > t_{19,1\%}$ e o valor da estatística caiu na região de rejeição, significando, portanto, que a nível de 1 % de significância rejeitamos a hipótese H_0 de que os algoritmos têm a mesma performance. Ao

contrário, verificamos que o teste acusa a superioridade do SISROTAU em relação ao algoritmo de MOLE & JAMESON.

b) SISROTAU X SAVING

De maneira análoga ao teste anterior, formemos as duas hipóteses básicas:

$H_0 : \bar{d} = 0$ (hipótese nula)

$H_1 : \bar{d} > 0$ (hipótese alternativa)

onde $\bar{d}$ é a média das diferenças d_i entre as soluções obtidas pelo SISROTAU e pelo método SAVING, nesta ordem. Em se confirmando a hipótese H_0 concluímos que os dois algoritmos em teste fornecem soluções equivalentes, caso contrário, o algoritmo SAVING tem um desempenho superior ao SISROTAU. Passando ao cálculo, teremos:

$$\sum_{i=1}^{20} d_i = 983 \qquad \text{e} \qquad \bar{d} = 49,1500$$

$$\sum_{i=1}^{20} d_i^2 = 259.000,0000$$

Desvio padrão da amostra:

$$s_d^2 = \frac{259.000,0000 - \frac{(983)^2}{20}}{19}$$

$$S_d = 105,3030$$

Valor da estatística:

$$t = \frac{49,15}{105,3030 \; / \; \sqrt{20}}$$

$$\boxed{t = 2,0873}$$

Uma vez que $t < t_{19,1\%} = 2,539$, o valor da estatística caiu dentro da região de aceitação e, portanto, não podemos rejeitar a hipótese H_0 de que os algoritmos têm a mesma performance.

Acreditando que o SISROTAU tem a sua melhor perfomance para problemas do tipo multi-praça-sede e, em especial, para um número elevado de praças-sedes, fomos à literatura à procura de problemas testes para compararmos o desempenho do SISROTAU nesta situação. Não encontramos na Bibliografia os dados de tais problemas ou, quando encontramos, referiam-se a PRVM que possuiam restrições adicionais ao modelo que estabelecemos no capítulo III e a comparação ficaria, portanto, prejudicada. Para contornar tal problema e compararmos a performance do SISROTAU em problemas do tipo multi-praça-sede tomamos o algoritmo sequencial de Mole & Jameson, modificando-o no sentido de resolver problemas deste tipo. A modificação é bastante simples: no procedimento original do algoritmo de Mole & Jameson as rotas eram inicializadas tomando-se o nó mais distante da praça-sede. Modificamos o algoritmo para construir as rotas somente pelo critério do menor esforço modificado (MESF), definido no capítulo III, seção III.3.1.3. Tomamos o algoritmo de Mole & Jameson original e fixamos o parâmetro λ (lambda) em zero. O primeiro nó a ser incluído na rota é aquele mais próximo à praça-sede que dá origem à

rota, os demais são incluídos pelo critério de MESF até que TMAXROT não permita nenhuma inclusão. O algoritmo teria, portanto, um funcionamento semelhante ao método de inserção do vizinho mais próximo (IMP) apresentado na seção do Caixeiro Viajante, com a diferença, óbvia, de que aqui temos M rotas a serem formadas contra apenas uma no caso do PCV. Designemos, ainda, tal algoritmo por MOLEMODI.

Uma vez que temos uma versão modificada para o o algoritmo sequencial de Mole & Jameson, geramos, aleatóriamente, vinte problemas do tipo multi-praça-sede, cada um deles com dez auditores/praças-sedes distintas e setenta nós

NUM.	MOLEMODI	SISROTAU	MOLEMODI/SISROTAU
1	5160(14)	3958(46)	1,3037
2	3889(14)	2896(42)	1,3429
3	5025(14)	3561(36)	1,4111
4	5402(14)	3307(52)	1,6335
5	3831(15)	3428(44)	1,1176
6	3543(14)	2900(36)	1,2217
7	3915(14)	3912(40)	1,0008
8	4592(15)	3078(40)	1,4919
9	4294(15)	4305(49)	0,9974
10	4436(14)	3616(36)	1,2268
11	4372(14)	4328(41)	1,0102
12	4148(14)	3446(40)	1,2037
13	4194(15)	3176(46)	1,3205
14	3414(15)	2772(45)	1,2316
15	4198(17)	4230(55)	0,9924
16	4713(14)	3536(55)	1,3329
17	3731(16)	2956(41)	1,2622
18	3775(15)	3108(38)	1,2146
19	3763(15)	3542(54)	1,0624
20	3447(15)	3297(47)	1,0455

QUADRO IV.10 - Problemas tipo PRVM

(agências). Não consideramos as restrições de impedimento e missão conjunta. Os vinte problemas foram gerados de forma semelhante àqueles do tipo PRV1 correspondentes ao quadro IV.9, com a diferença de que aqui geramos aleatóriamente, também, as coordenadas dos nós que representam os depósitos (ou praças-sedes).

O quadro IV.10 apresenta as soluções obtidas pelos dois algoritmos em teste, os tempos de processamento em um microcomputador PC-XT e a relação entre as soluções obtidas pelo MOLEMODI e SISROTAU, nesta ordem. Não se faz necessária a aplicação do teste de hipótese para concluirmos que o SISROTAU produz soluções consideravelmente melhores do que o outro algoritmo em teste. No caso do quadro IV.10 a média desta vantagem, para os vinte problemas, atinge a casa dos 22 % (vinte e dois porcento). A superioridade do SISROTAU reside no fato de que este considera, através da aplicação do procedimento AGRUPA, uma 'competição' entre as diversas praças-sedes, agrupando antes de sequenciar os diversos nós.

Antes de apresentarmos mais outro teste para o algoritmo SISROTAU precisamos tecer alguns comentários sobre a base de dados utilizada em situações reais. Nos exemplos mostrados anteriormente trabalhamos com as coordenadas retangulares dos nós para a localização e cálculo de distâncias entre estes mesmos nós. As distâncias são, portanto, calculados na norma euclideana, correspondendo ao comprimento do segmento de reta que une dois nós quaisquer. Tais distâncias, embora não correspondam às distâncias reais da rede rodoviária que liga as diversas agências e praças-sedes, pode ser usada para aproximar aquelas.

Assad [3] apresenta várias formas de se trabalhar com os dados geográficos de uma rede para um sistema de roteamento. O autor cita, por

exemplo, a possibilidade de tomarmos parâmetros para corrigir as distâncias euclideanas, baseados em características regionais da rede rodoviária, aproximando-as ainda mais das distâncias reais. A terceira forma possível é a de trabalharmos com distâncias reais e uma matriz de caminhos mínimos. Em se trabalhando com distâncias reais teríamos que armazenar alguns arcos (rodovias que ligam dois pontos da rede) da rede e a toda vez que fossemos resolver um determinado problema de roteamento teríamos que, antes, resolver um problema de caminho mínimo, utilizando, por exemplo, o algoritmo de Djkstra (Syslo [34]). Uma vez resolvido o problema de caminhos mínimos teríamos, para cada par de nós, a distância real mínima entre eles. É interessante observar, ainda, que, em se trabalhando com matriz de caminhos mínimos, haveria necessidade de se trabalhar com um número maior de nós do que aqueles que se pretende rotear. Tal situação acontece porque estaríamos trabalhando, a cada período de roteamento, com algumas das agências (nós) e não a totalidade delas. Sendo assim, o processo de cálculo de caminhos mínimos poderia necessitar de nós intermediários que, apesar de não serem candidatos a roteamento, definem o caminho mínimo entre dois pares de nós a serem roteados. Evidentemente, estamos supondo o cálculo dos caminhos mínimos a cada período de roteamento, uma vez que o armazenamento dos valores de caminhos mínimos para todos os pares de nós seria extremamente dispendioso.

No presente trabalho nós utilizamos as coordenadas retangulares, X e Y dos pontos da rede para calcular as distâncias entre os diversos pares de nós. Embora pudéssemos calcular as distâncias a cada instante do algoritmo, preferimos construir uma estrutura de dados do tipo ponteiro (TURBO PASCAL 4.0) para armazenar as distâncias, préviamente calculadas, levando em conta a simetria

do problema e armazenando, tão somente, a matriz triangular superior de distâncias. Para a rede de agências do Banco do Brasil, conseguimos as coordenadas geográficas junto ao IBGE-RJ (Instituto Brasileiro de Geografia e Estatística). A partir destas coordenadas, latitude e longitude, utilizamos fórmulas de projeção cartográfica, fórmulas de Puissant, para obtermos as coordenadas retangulares, X e Y, de cada nó.

Coletamos na PRESI-AUDIT, órgão da Direção Geral do Banco do Brasil, os dados relativos a dois períodos de roteamento e utilizamos o SISROTAU para resolvê-los e comparar os resultados. No quadro

	PR1	PR2
Número de Auditores	84	77
Número de agencias de missão simples	196	164
Número de agências de missão conjunta	23	32

Quadro IV.11 - problemas reais

IV.11 apresentamos as características dos problemas PR1 e PR2, problemas reais, nos quais algumas das agências do Banco foram selecionadas para serem roteadas. A seleção destas agências dentre as mais de quatro mil existentes, como já foi mencionado, se deve ao fato de que, dentro dos respectivos períodos, elas necessitavam de auditoria e as outras não. No quadro IV.12 apresentamos as soluções obtidas pelo método de roteamento usualmente adotado no Banco, ao qual denominaremos PROCESSAMENTO MANUAL, e as soluções obtidas pelo nosso algoritmo SISROTAU. Para os dois problemas, o

QUADRO IV.12 - Problemas PR1 e PR2

RESULTADOS OBTIDOS PELO ROTEAMENTO MANUAL

	PR1	PR2
Distancia total de roteamento(em Km)	62.612	49.499
Tempo processamento	(*)	(*)
Número de auditores utilizados	84	77

RESULTADOS OBTIDOS PELO SISROTAU

	PR1	PR2
Distância total de roteamento (em Km)	49.634	36.697
Tempo processamento	20 min	20 min
Número de auditores utilizados	79	71

(*) - processamento manual - não avaliado

SISROTAU obteve uma solução melhor, no PR1 a distância total percorrida foi cerca de 20 % menor e para o PR2 a economia atingiu os 25 % . Notamos, ainda, que o SISROTAU utilizou uma quantidade inferior de auditores para efetuar os dois roteamentos. No PR1 economizou 5 auditores, no PR2 economizou 6 auditores. O PROCESSAMENTO MANUAL do roteamento de auditores não é feito, em geral, de maneira contínua como no caso do SISROTAU e, portanto, não avaliamos o tempo para os dois problemas, PR1 e PR2. Tal tempo é, muitas vezes, considerado em dias. No ANEXO II listamos os dados para o PR1 e PR2 e a solução gerada pelo SISROTAU.

Com relação aos problemas do tipo PRVM, a geração aleatória é feita de forma a simular os problemas reais de Auditores do Banco do Brasil. Aqui, para não haver dúvidas, estamos tratando do PRVM, considerando a já citada analogia com o PRA.

CAPÍTULO V

CONCLUSÄO

Acreditamos ter desenvolvido um algoritmo bastante flexível no sentido de resolver tanto o Problema de Roteamento 1-depósito, quanto o Problema de Roteamento multi-depósito. No primeiro caso, os resultados estiveram no mesmo nível daqueles obtidos por algoritmos clássicos da literatura. No segundo, caso multi-depósito, obtivemos resultados consideravelmente melhores nos testes que realizamos. Uma vez que a literatura não oferece uma quantidade razoável de bons algoritmos para o caso multi-depósito, acreditamos que, a nível teórico, aí reside a nossa principal contribuição.

De outro lado, acreditamos ter fornecido um modelo matemático e o correspondente algoritmo para a racionalização dos custos com os serviços de auditoria do Banco do Brasil S.A. Os testes realizados com problemas reais, PR1 e PR2, mostraram que o SISROTAU pode reduzir o valor da distância total percorrida ao longo de um período de roteamento, quando comparada com as soluções obtidas pelo procedimento atualmente utilizado no Banco. Citamos, também, o fato de que a implementação do SISROTAU promoveria uma acentuada agilização dos serviços de roteamento dos auditores, podendo, porconseguinte, liberar mão de obra para outros serviços. É interessante observar, também, que o SISROTAU foi desenvolvido para um microcomputador do tipo PC-XT, cujo custo de processamento é imcomparavelmente menor do que o custo de processamento em sistemas de grande porte. Com algum custo adicional poderíamos colocar o SISROTAU para funcionar em um microcomputador mais

sofisticado, por exemplo um PC-AT, e os tempos de processamento poderiam ser reduzidos em até cinco vezes.

Finalmente, gostaríamos de tecer alguns comentários práticos acerca da implementação do SISROTAU. Nós acreditamos que um bom sistema de roteamento deve ser flexível no sentido de permitir a iteração Homem/máquina e o ajuste dinâmico das rotas. Abordamos um sistema de roteamento que considera as rotas estáticas dentro de um período de planejamento. Na realidade, acontecimentos casuais podem impedir que algumas destas rotas sejam concretizadas e, então, necessáriamente, deverão sofrer um ajuste para que o sistema flua sem qualquer prejuízo. Há, portanto, que se somar ao nosso trabalho algumas rotinas que, dinamicamente, promovam tais ajustes. De outro lado, acreditamos que uma iteração Homem/máquina através de uma rotina gráfica pode refinar a solução de forma proveitosa e, às vezes, em menor tempo que uma rotina automática. Tais rotinas exigem, porém, equipamento mais sofisticado a nível de hardware, o que envolve um maior investimento no sistema.

REFERÊNCIAS BIBLIOGRÁFICAS

1- ALMEIDA,MARIA TERESA -"Problema de Distribuição e suas extensões - uma revisão bibliográfica", doc.de trabalho n. 46, Instituto Superior de Economia Universidade Técnica de Lisboa, Junho/1988.

2- ANDEBERG,M.R. - Cluster Analysis for Applications, ACADEMIC PRESS, 1973.

3- ASSAD, A.A. - " Modeling and Implementation issues in Vehicle Routing" in VEHICLE ROUTING METHODS AND STUDIES edited by Bruce L.Golden and Arjang A. Assad, NORTH-HOLLAND, 1988.

4- BALAS, E. & TOTH,P.- "Branch and Bound Methods" in The Travelling Salesman Problem edited by E.L.Lawler and J.K.Lenstra, John Wiley (1985).

5- BODIN, L., GOLDEN,B., ASSAD,A. e BALL,M. - "Routing and Scheduling of Vehicles and Crews - the state of art ", Comput & Ops.Res. vol 10, n.2, pp 63-211, 1983.

6- BORNSTEIN,C.T. & GALDINO,G.P.J - "Complexidade de Algoritmos e Problemas NP-Completos em Otimização Combinatória", Relatório Didático ES-25/83, COPPE/UFRJ - 1983

7- CASSIDY,P.J. & BENNETT, H.S. - " TRAMP - A Multi-Terminal Vehicle Scheduling System ", OPERATIONAL RESEARCH QUARTERLY, vol.23, n.2, 1972.

8- CLARKE,G. & WRIGHT, J.W. - " Scheduling of Vehicles from a central depot to a number of delivery points",OPERS.RES., vol 12, n.4, 1964.

9- CHRISTOFIDES, N.- "Worst-Case Analysis of a new heuristc for the Travelling Salesman Problem", Report 388. Graduate Scholl of Industrial Administration Carnegie Mellon University (1976).

10- CHRISTOFIDES,N. & EILON,S. - "An algorithm for the Vehicle-Dispatching Problem", OPERATIONAL RESEARCH QUARTERLY, vol 20, n.3, 1969.

11- CHRISTOFIDES, NICOS - Graph Theory - An Algorithmic Approach, Academic Press (1975).

12- CHRISTOFIDES,N., MINGOZZI,A. e TOTH,P. - " The Vehicle Routing Problem " in Combinatorial Optimization editado por Christofides, N., Mingozzi,A., Toth, P. e Sandi,C. - John Wiley & Sons (1979)

13- CHRISTOFIDES, N. - " The Travelling Salesman Problem" in Combinatorial Optimization, editado por Christofides,N., Mingozzi,A., Toth,P. e Sandi,C. - John Wiley & Sons (1979).

14- DIDAY,E. & SIMON,J.C. - Clustering Analysis - Communications and cybernetics, HEIDELBERG, 10,1976.

15- DURAN,B.S. & ODELL,P.L. - Cluster Analysis - a survey, HEIDELBERG SPRINGER - VERLAG BERLIN, 1974.

16- FORGY,E.W. -"Cluster Analysis of Multivariate Data: efficiency versus interpretability of classifications", BIOMETRICS, 21(3):768, 1965.

17- GASKELL,T. - " Bases for Vehicle Fleet Scheduling", OPERATIONAL RESEARCH QUARTERLY, vol.18, 1967.

18- GILLETT,B.E. & MILLER,L.R. - "A heuristic algorithm for the Vehicle Dispatch Problem", OPS.RES, 22, pp 340-349, 1974.

19- GILLETT,B.E. & JOHNSON,J.E. - " Multi-terminal Vehicle-Dispatch Algorithm", OMEGA 4, 1976.

20- GOLDEN, BRUCE L. & ASSAD, ARJANG A. - Vehicle Routing. Methods and Studies, North-Holland - 1988

21- GOLDEN, B.L. & STEWART, W.R. - "Empirical Analysis of Heuristics" in The Traveling Salesman Problem pp 207-244, John Wiley & Sons 1985.

22- GOLDEN, B.L.,MAGNANTI,T. e NGUYEN,H. - "Implementing Vehicle Routing Algorithms", NETWORKS 7, pp 113-148, 1977.

23- HARTIGAN, J.A - Clustering Algorithms, New York, JOHN WILEY & SONS, 1971

24- HOFFMAN, A.J & WOLFE, P. - " History the Travelling Salesman Problem" in The Travelling Salesman Problem editado por E.L.Lawler, J.K.Lenstra, John Wiley & Sons ltd. (1985).

25- LENSTRA, J. & KAN, A.RINNOOY - " Complexity of Vehicle Routing and Scheduling Problems", Networks (1981)

26- LIN, S. & KERNIGHAN,B.W. - "An effective heuristic algorithm for the Travelling Salesman Problem", OPER.RES. 21 (1973).

27- LUCAS,L.C.DE SÁ, - Análise de Grupamentos, Dissertação de Mestrado em Engenharia de Sistemas, COPPE/UFRJ, 1983.

28- MOLE,R.H. & JAMESON,S.R. - " A sequential route-building algorithm employing a generalised saving criterion", OPERATIONAL RESEARCH QUARTERLY, vol 27, n.2, 1976.

29- PADBERG, M. & SUNG, T.Y. - "An Analytical Comparison of Different Formulations of the TSP", R & T 41663 Stern School of Business New York University (1988)

30- PAPADIMITRIOU, C.H. & JOHNSON, D.S. - "Performance guarantees for heuristics" in The Travelling Salesman Problem edited by E.L.Lawler and J.K.Lenstra, John Wiley (1985).

31- PINHO GAMA, M. - Bases da Análise de Grupamento, UNIVERSIDADE DE BRASÍLIA, INSTITUTO DE CIÊNCIAS EXATAS, DEPARTAMENTO DE ESTATÍSTICA, 1980.

32- SOLOMON, MARIUS M. - "On then Worst-Case Performance of Some Heuristics for the Vehicle Routing and Scheduling Problem with Time Window constraints", Networks vol 16 (1986) 161-174.

33- STEVENSON, WILLIAN J. - Estatística Aplicada à Administração - Ed. Harbra (1981).

34- SYSLO,MACIEJ M., - Discrete Optimization Algorithms with Pascal Programs , Prentice-Hall (1983).

35- TILLMAN,F. & CAIN T. - "An upper bounding algorithm for the Single and Multiple Terminal Delivery Problem", MANAGMENT SCIENCE, vol.18, n.11, 1972.

36- VELOSO,P., SANTOS,C., AZEREDO,P.,FURTADO,A. - Estrutura de Dados, Editora Campus, 1986.

37- WREN,A. & HOLLIDAY,A. - "Computer Scheduling of Vehicles from one or more depots to a number of delivery points", OPERATIONAL RESEARCH QUARTERLY, vol 23, n.3, 1972.

38- YELLOW,P.- "A computational modification to the Savings method of Vehicle Scheduling", OPERATIONAL RESEARCH QUARTERLY, vol 21, 1970.

A N E X O I

I - Testes de Hipóteses

Frequentemente, estamos interessados em conhecer alguns parâmetros de uma população, como média, desvio padrão, variância e outros mais. Realizar um censo, para uma população finita ou infinita, é por demais trabalhoso ou mesmo impossível e, então, recorremos à amostragem da população. A partir de uma amostra poderemos inferir estatísticas que se aproximarão dos verdadeiros parâmetros da população. A questão natural que surge é: quão próxima está a estatística amostral do verdadeiro parâmetro populacional ? Para responder a esta pergunta teremos de recorrer a uma distribuição amostral daquela estatística sobre a qual queremos inferir. Iremos, agora, apresentar alguns conceitos fundamentais em estatística que podem ser vistos de maneira mais detalhada em Stevenson [33].

Definição : Uma distribuição amostral é uma distribuição de probabilidades que indica até que ponto uma estatística amostral tende a variar devido a variações casuais na amostragem aleatória.

Teorema do Limite Central:

1- Se a população sob amostragem tem distribuição normal, a distribuição das médias amostrais também será normal para todos os tamanhos de amostra.

2- Se a população básica é não-normal, a distribuição de médias amostrais será

aproximadamente normal para grandes amostras.
Distribuição Amostral de Médias: Quando estamos realizando amostragem no sentido de inferir sobre a média de uma população, a estatística nos fornece os seguintes resultados:

1- A média de uma distribuição amostral, $\upsilon(\bar{x})$, é sempre igual à média da população, $\upsilon(x)$.

2- Quando a população é muito grande ou infinita, o desvio padrão da distribuição amostral de média, $\sigma(\bar{x})$, guarda a seguinte relação com o desvio padrão da população ($\sigma(x)$): $\sigma(\bar{x}) = \dfrac{\sigma(x)}{\sqrt{n}}$

O mesmo autor, Stevenson, afirma, ainda, que uma regra prática muito usada é a de que amostras que contenham mais de trinta observações terão uma distribuição amostral de médias aproximadamente normal.

De posse destes conceitos iniciais, passaremos a descrever os testes de hípoteses em estatística. Seja H_0 uma hipótese existente acerca de determinada população, denominada hipótese nula, e H_1 a hipótese alternativa. Tomamos uma amostra da população e iremos testar a veracidade da hipótese nula. O teste irá levar-nos à aceitação ou rejeição da hipótese H_0. Tomemos um exemplo:

Suponhamos que um certo pesquisador de renome apareceu em um importante congresso científico com um novo algoritmo heurístico para resolver o Problema do Caixeiro Viajante. O referido pesquisador disse que o seu algoritmo fornecia soluções com um desvio médio de 5 % em relação ao ótimo para toda a população de problemas possíveis de Caixeiro Viajante. Forneceu, ainda, o desvio padrão da população, 2 % . Um estudante curioso

resolveu testar a veracidade da afirmação do referido pesquisador e gerou, aleatóriamente, quarenta problemas de Caixeiro Viajante, obtendo as soluções por um algoritmo exato e as soluções heurísticas com o algoritmo em teste. O desvio médio observado pelo estudante foi de 7 % acima do ótimo. O que o estudante poderia concluir ?
Designemos o desvio médio da população e o seu desvio padrão por $\upsilon(x)$ e $\sigma(x)$ respectivamente. A distribuição amostral das médias será caracterizada pelos parâmetros:

$$\sigma_{(\bar{x})} = \frac{2}{\sqrt{40}} = 0,3162$$

$$\upsilon_{(\bar{x})} = 5$$

No gráfico II.1, marcamos o valor da média encontrada pelo estudante, 7 % , correspondendo à variável reduzida z = 6.325. A probabilidade α de se obter uma amostra com desvio médio superior a 7 % é praticamente zero, pois a área indicada por α na figura é praticamente nula. O estudante poderia concluir, portanto, que alguma coisa deve estar errada na afirmação do pesquisador.

No gráfico II.2 verificamos que o valor Z = 1.65, reduzido, é um marco fundamental, "valor crítico" para a nossa decisão de aceitar ou rejeitar uma determinada hipótese H_0 A área hachurada, numericamente igual a α, representa a probabilidade de rejeitarmos a hipótese H_0 quando ela é verdadeira. O valor α é denominado nível de significância do teste. Então, todo teste de hipótese consiste em se determinar a priori o valor α, ou seja, o risco que se quer correr de

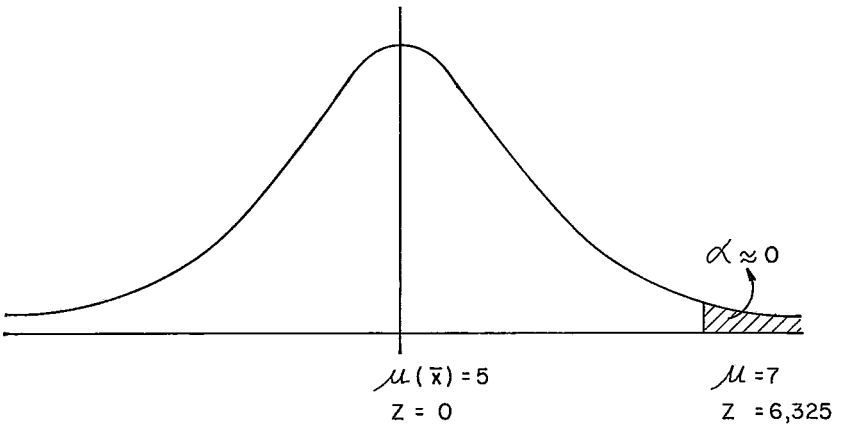


FIG. II. 1

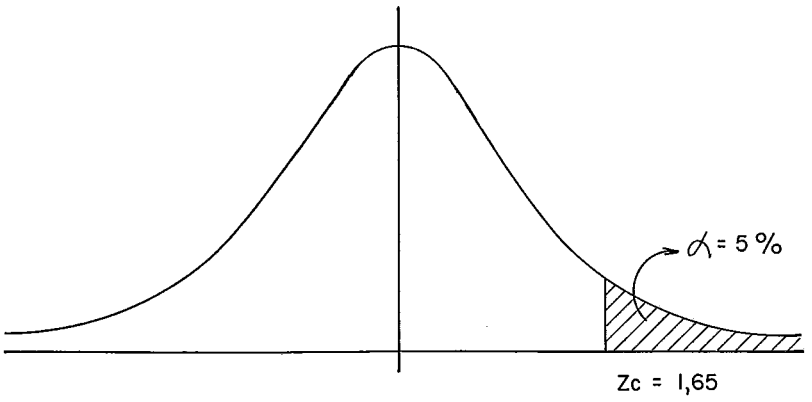


FIG. II. 2

rejeitar a hipótese H_0 quando ela é verdadeira. Tomando o exemplo anterior, se o estudante obtivesse um desvio médio de 5.5 % ao invés de 7 % e estivéssemos realizando o teste ao nível de 1 % de significância teríamos:

$$Z = \frac{(5.5 - 5)}{0.3162} = 1.581$$

Recorrendo à tabela que fornece os valores das áreas sob a distribuição normal, entrando com o valor de $\alpha = 0.01$, determinamos o valor crítico:

Z_c (valor crítico) = 2.33

Como o valor Z = 1.581 da estatística em questão é menor do que o valor crítico, o estudante não rejeitaria a hípotese testada. Diríamos, portanto, que a evidência amostral não foi significativa, ao nível de 1 % , para rejeitar a hipótese (vide figura II.3). No exemplo acima fizemos um teste unilateral, pois averiguamos apenas a possibilidade da média amostral ser superior ao valor colocado em hipótese.
A figura II.4-a representa um teste bilateral ao nível de significância α. As figuras II.4-b e II.4-c representam testes unilaterais.

Nos testes de hipótese podemos cometer dois tipos de erros:

erro tipo I : quando rejeitamos a hipótese H_0 sendo ela verdadeira. A probabilidade de cometer este tipo de erro é α .

erro tipo II: quando aceitamos a hipótese H_0 sendo ela falsa. Denominaremos β a probabilidade de cometer este tipo de erro.

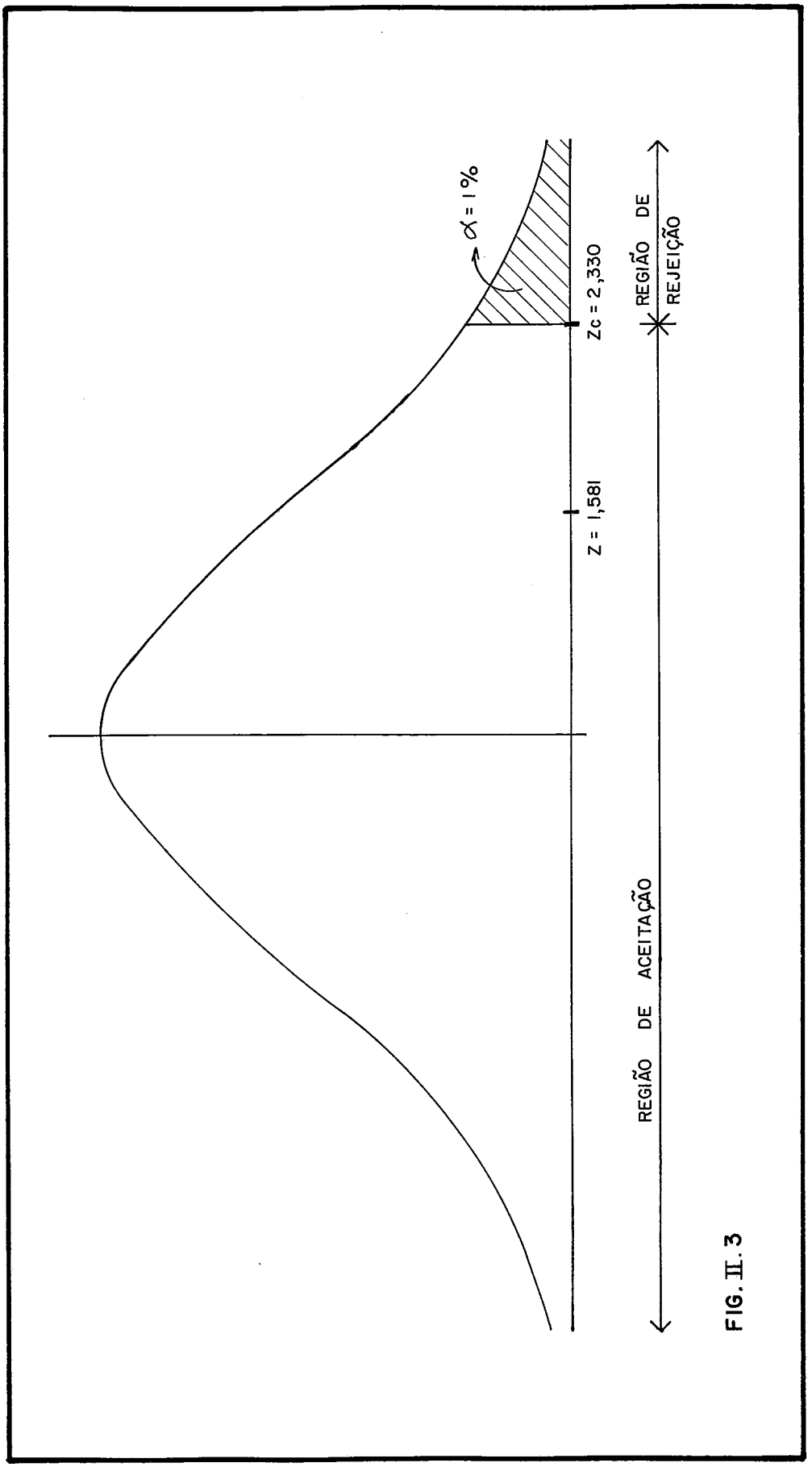


FIG. II. 3

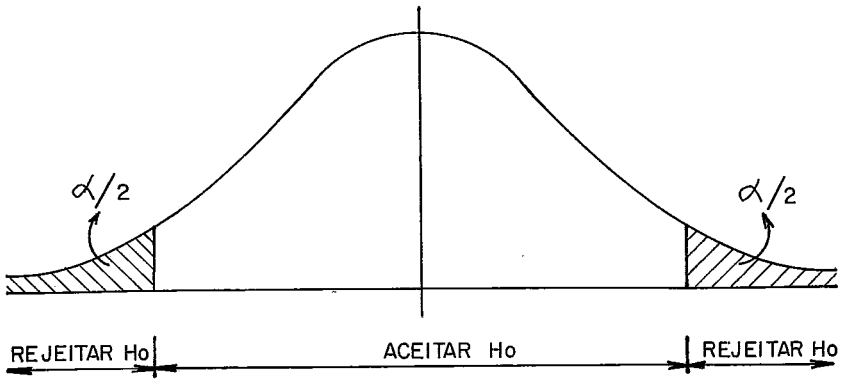


FIG. II. 4a

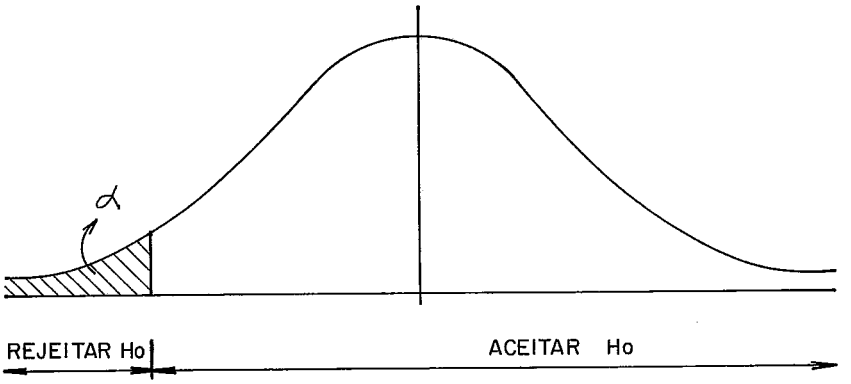


FIG. II. 4b

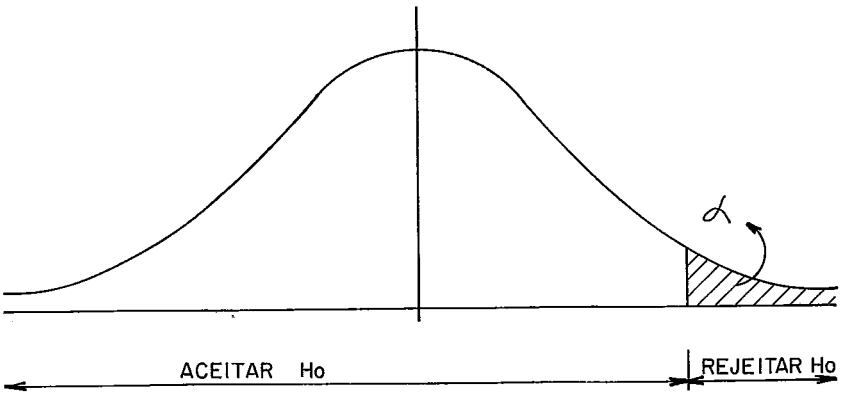


FIG. II. 4c

II - Testes de Hipóteses para Médias

Consideraremos, agora, o caso em que temos duas ou mais amostras, em princípio de populações distintas. Com base nessas amostras, iremos comparar parâmetros equivalentes das populações envolvidas. Basicamente, este tipo de teste é aquele que mais nos interessará, quando estivermos interessados em comparar os resultados fornecidos por duas heurísticas distintas, colocadas em competição. Em termos gerais, testamos hipóteses referentes ao valor real da diferença entre duas médias populacionais:

$$H_0 : \upsilon_1 - \upsilon_2 = \Delta$$

Tendo, em geral, especial interesse o caso em que $\Delta = 0$, em que se testa a hipótese da igualdade das duas médias, ou seja, $\upsilon_1 = \upsilon_2$. Consideremos apenas o caso de dados emparelhados. Dizemos que os dados estão emparelhados quando eles estão relacionados dois a dois segundo algum critério. Assim, por exemplo, quando geramos aleatóriamente problemas de Caixeiro viajante e resolvemo-los, um a um, por duas heurísticas distintas, os resultados obtidos por ambas as heurísticas, e para cada problema, estão emparelhados dois a dois. Uma influência individual de determinado problema na qualidade da solução seria sofrida igualmente pelos dois algoritmos e daí desprezamos tais influências individuais. Se os dados das duas amostras estão emparelhados, tem sentido calcularmos as diferenças d_i correspondentes a cada par de valores, reduzindo-se os dados a uma única amostra de n diferenças. Por outro lado, testar a hipótese de que a diferença entre as médias das duas populações emparelhadas seja igual a um certo valor Δ,

equivale testar a hipótese de que a média de todas as diferenças seja igual a Δ, o que decorre diretamente das propriedades da média. Então, vamos testar simplesmente a hipótese:

H_0: $\upsilon_d = 0$

contra uma alternativa H_1 que poderá corresponder a um teste unilateral, por exemplo, $\upsilon_d < \Delta$, ou a um teste bilateral, $\upsilon_d \neq \Delta$, o que equivale a $\upsilon_d < \Delta$ e $\upsilon_d > \Delta$. É interessante observar que recaímos no teste de uma única média conforme exemplificamos no ítem anterior, com a diferença de que não conhecemos o desvio padrão da população, σ , e, portanto, não utilizaremos a distribuição normal para a variável aleatória υ_d , mas sim a distribuição t de Student com n-1 graus de liberdade. A distribuição t de Student é aproximadamente igual à distribuição normal e mais indicada nos casos em que não se conhece o real valor do desvio padrão da população, sendo este substituído pelo desvio padrão da amostra aleatória que passaremos a denominar s_d. Assim calculamos:

$$t = \frac{\bar{d} - \Delta}{s_d / \sqrt{n}}$$

onde

$\bar{d}$ é a média da amostra das diferenças
Δ o valor testado da média das diferenças nas populações
s_d o desvio padrão da amostra das diferenças
n o tamanho da amostra das diferenças

Para ficar claro o teste das diferenças de médias, tomemos mais um exemplo:
Consideremos que dois algoritmos heurísticos, A e B, estão sendo testados com 30 problemas gerados aleatóriamente. Supomos conhecidas as soluções exatas para todos os problemas testes e desejamos comparar a eficiência dos dois algoritmos no sentido de indicar aquele que produz soluções mais próximas das respectivas soluções ótimas. A tabela II.5 mostra os resultados obtidos no teste, ou seja, os desvios em relação ao ótimo para cada problema e para ambos os algoritmos em teste. Consideremos aproximadamente normal a distribuição amostral das diferenças de desvios, d = desvio A - desvio B e formulemos a seguinte hipótese:

$H_0 : \overline{d} = \Delta = 0$

onde $\overline{d}$ é a média das diferenças de desvios que estamos supondo ser nula. Estamos testando na verdade a hipótese dos dois algoritmos, A e B, produzirem desvios iguais para a população de problemas aos quais ambos são destinados.

Tomemos, agora, a hipótese alternativa:

$H_1 : \overline{d} < 0$

A nossa hipótese alternativa corresponde a um teste unilateral. Verificamos, portanto, a hipótese alternativa de o algoritmo A produzir desvios menores do que o algoritmo B. Tomemos, ainda, o nível $\alpha = 5$ % de significância para realizarmos o teste. Passemos aos cálculos:

Num.	desv.A(%)	desv.B(%)	(A - B)
1	10	10	0
2	9	9	0
3	10	13	-3
4	11	14	-3
5	10	15	-5
6	11	11	0
7	10	17	-7
8	11	13	-2
9	12	11	1
10	10	14	-4
11	12	12	0
12	8	15	-7
13	9	9	0
14	12	10	2
15	9	13	-4
16	10	18	-8
17	10	15	-5
18	10	10	0
19	12	13	-1
20	10	9	1
21	10	8	2
22	10	12	-2
23	11	18	-7
24	10	9	1
25	9	8	1
26	10	11	-1
27	10	15	-5
28	11	14	-3
29	10	10	0
30	10	10	0

Tabela II.5 (teste dos algoritmos A e B)

$$t = \frac{\bar{d} - \Delta}{s / \sqrt{n}}$$

$$\bar{d} = \frac{\sum_{i=1}^{30} d_i}{30} = \frac{-59}{30} = -1,96$$

$$s_d^2 = \frac{\sum_{i=1}^{30} d_i^2 - \frac{\left(\sum_{i=1}^{30} d_i\right)^2}{30}}{29}$$

$$s_d = 2.94$$

$$t = -3.65$$

Recorremos à tabela da distribuição t de Student para calcularmos o valor crítico. Entramos com ϑ = 29 (grau de liberdade = n - 1) e α = 0.05 (5 %) e teremos:

valor crítico = $t_{29,5\%} = -2.756$

notamos, portanto, que a nossa estatística t caiu na região de rejeição (vide fig. II.6) e, então, rejeitaremos a hipótese H_0 ao nível de 5 % de significância. O teste nos aponta, portanto, que o algoritmo A produz soluções melhores do que o algoritmo B e não soluções iguais como previa a hipótese H_0 .

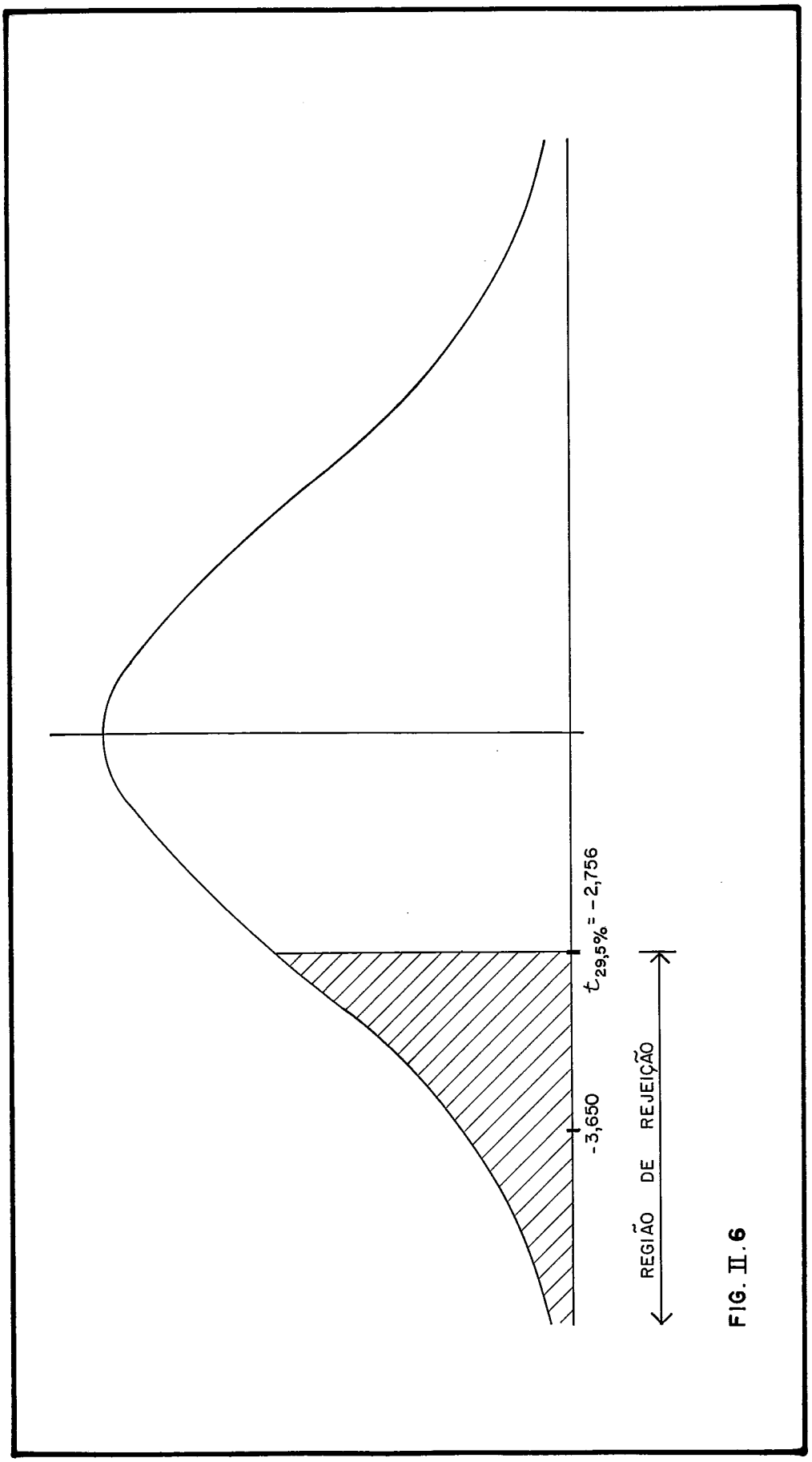


FIG. II.6

É interessante observar que não podemos garantir com certeza absoluta que o algoritmo A produz soluções melhores do que B, posto que poderíamos ter cometido o erro do tipo I no teste acima, ou seja, rejeitamos a hipótese H_0, quando na verdade ela era verdadeira e os algoritmos teriam basicamente a mesma performance. Ainda tentando esclarecer, uma explicação bastante simples seria a de que tivemos a infelicidade de tomar uma amostra cujos resultados nos conduziram exatamente naquela faixa de risco que assumimos quando determinamos o valor de $\alpha = 5\ \%$. Não custa lembrar que o presente teste supõe que os dados estejam emparelhados e, portanto, as duas amostras tem o mesmo tamanho. Ademais, no caso de amostra com mais de 30 observações poderíamos utilizar uma distribuição normal para a média, embora a distribuição t de Student seja, teóricamente, mais correta.

III - Testes Não-Paramétricos

No ítem anterior testamos hipóteses referente à média, que é um parâmetro populacional, ou à diferença entre médias de duas amostras. Agora, abordaremos testes que se referem a outros aspectos que não os parâmetros em si. Apresentaremos, um tipo de teste, não-paramétrico, independente da forma da distribuição da população e do tamanho da amostra em estudo, o teste de Wilcoxon-Mann-Whitney. Este teste fornece uma alternativa interessante para a comparação de duas populações, sendo baseado na soma de "postos" dos valores observados. O posto de um valor, em um conjunto de n valores, é um número que indica sua posição no conjunto ordenado, crescente ou decrescente, do primeiro ao enésimo.

Wilcoxon considerou que, sendo válida a hipótese H_0 de identicidade entre as populações, as somas dos postos nas amostras deveriam fornecer valores intermediários compatíveis com os tamanhos de cada amostra. Poderíamos, portanto, determinar quais os limites para a soma dos postos nas amostras além dos quais devemos rejeitar H_0 . Mann e Whitney, entretanto, desenvolveram um procedimento mais adequando para o teste, baseado no cálculo de qualquer uma das quantidades:

$$\upsilon_1 = n_1 \cdot n_2 + \frac{n_1 \cdot (n_1 + 1)}{2} - T_1$$

$$\upsilon_2 = n_1 \cdot n_2 + \frac{n_2 \cdot (n_2 + 1)}{2} - T_2$$

onde n_1 e n_2 são os tamanhos das duas amostras e T_1 e T_2 as respectivas somas dos postos. Existem tabelas para a realização do teste com base em υ_1 ou υ_2. Entretanto, para $n_1 > 7$ e $n_2 > 7$, o teste pode ser realizado por aproximação pela normal, sendo que, para H_0 verdadeira, temos:

$$\upsilon(\upsilon_1) = \upsilon(\upsilon_2) = \frac{n_1 \cdot n_2}{2}$$

$$\sigma(\upsilon_1) = \sigma(\upsilon_2) = \sqrt{(n_1 \cdot n_2 \cdot (n_1 + n_2 + 1)) / 12}$$

O teste pode também ser aplicado ao caso de dados emparelhados. Para tanto, calcula-se as diferenças d_i entre os valores das duas amostras e considera-se os postos de seus valores absolutos. Se as populações forem idênticas, a soma dos postos das diferenças positivas deverá ser aproximadamente igual á soma dos postos das diferenças negativas, caso contrário rejeita-se H_0. Existem tabelas que

dão os valores críticos das somas de postos, entretanto, para $n \geq 25$ a soma dos postos T das diferenças, positivas ou negativas, pode ser testada por aproximação normal, com

$$\upsilon (T) = \frac{n . (n + 1)}{4}$$

$$\sigma (T) = \sqrt{n . (n + 1) . (2n + 1) / 24}$$

Para o cálculos dos postos das diferenças tecemos as seguintes considerações:

1- Pares de valores com diferença nula devem ser excluídos do cálculo de T.

2- Havendo valores iguais, considerar-se-á um posto médio. Por exemplo, no conjunto formado pelos cinco valores { 15,12,16,19,16} os postos, considerada uma ordenação crescente, serão, respectivamente, 2, 1, 3.5, 5, 3.5. Voltemos ao exemplo representado pela tabela II.5 e apliquemos o teste de Wilcoxon, tomando um nível de significância $\alpha = 5\ \%$. Ordenando os desvios d_i da tabela II.5 e desprezando os oito valores nulos teremos:

Num. do problema	desvio	posto
3	-3	12
4	-3	12
5	-5	17
7	-7	20
8	-2	8.5
9	1	3.5
10	-4	14.5
12	-7	20
14	2	8.5
15	-4	14.5
16	-8	22

17	-5	17
19	-1	3.5
20	1	3.5
21	2	8.5
22	-2	8.5
23	-7	20
24	1	3.5
25	1	3.5
26	-1	3.5
27	-5	17
28	-3	12

Colocamos valores médios para os desvios empatados em um mesmo valor. Assim, os postos para os desvios de valor absoluto igual a 1 foram calculados da seguinte forma:(1 + 2 + 3 + 4 + 5 + 6) / 6 = 3.5. O mesmo raciocínio se aplica para os demais empates que aparecem. Tomemos, agora, a soma dos postos para os desvios negativos:

T = (3 x 12 + 3 x 17 + 3 x 20 + 2 x 8.5 + 2 x 14.5 + 2 x 3.5 + 22) = 222

Tomando as fórmulas para dados emparelhados, citadas anteriormente, teremos:

$$\upsilon (T) = \frac{22 (22 + 1)}{4} = 126,50$$

$$\sigma (T) = \sqrt{(22 . 23 . 45)/24} = 30.80$$

Calculamos, portanto, a estatística

$$Z = \frac{222 - 126.5}{30.80} = 3.100$$

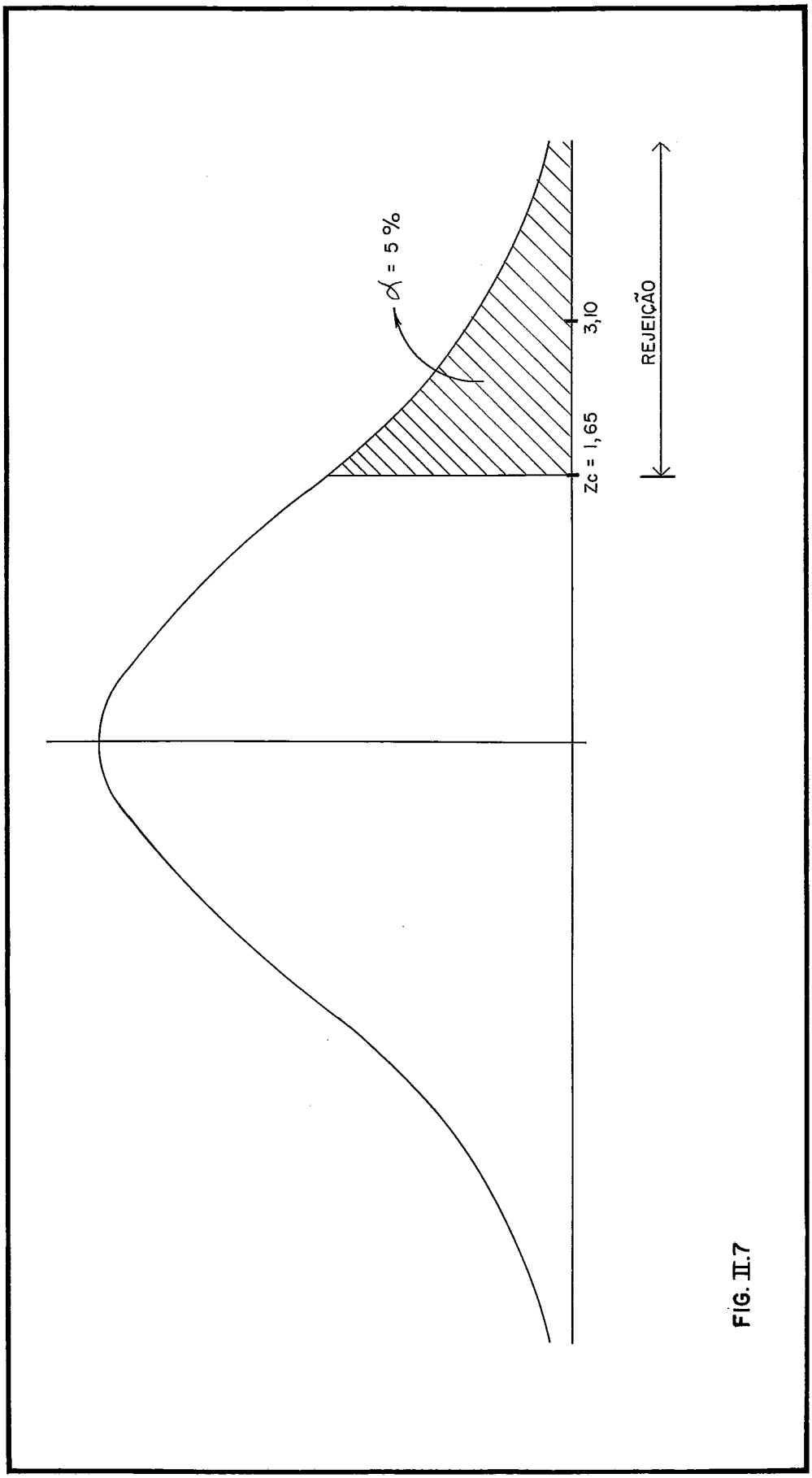


FIG. II.7

De posse do valor $\alpha = 0.05$ vamos à tabela para a distribuição normal e pegamos o valor crítico, $Z_c = 1.65$. Verificamos, portanto, que o valor da estatística Z é superior ao valor crítico, situando-se a mesma na região de rejeição conforme se pode ver no gráfico II.7

Evidentemente obtivemos o mesmo resultado do teste de diferença de médias anterior. A conclusão é a de que deveremos rejeitar a hipótese H_0 de que os algoritmos tem a mesma performance, e mais, verificamos que a soma dos postos das diferenças negativas foi maior do que os de diferença positiva. Na forma como foi colocada a hipótese alternativa, H_1, $\bar{d} < 0$, onde $\bar{d}$ é a média das diferenças $d_A - d_B$, concluimos que o algoritmo A fornece melhores soluções do que o algoritmo B.

A N E X O I I

Listamos neste ANEXO os dados correspondentes aos problemas reais apresentados no quadro IV.11 do capítulo IV. Listamos, também, as rotas geradas pelo SISROTAU e os parâmetros utilizados no processamento.

I - Problema Real Um (PR1)

a)-DIMENSÕES:
- número de auditores (M) = 84
- número de agências do tipo missão simples (NAS) = 196
- número de agências do tipo missão conjunta = 23

b)- PARÂMETROS:
- LAMBDA = 1.5
- MI = 2.5
- TMAXROT = 72 dias
- TMAXREFIN = 15 min
- DISTMAX = 1500 Km
- RMAX = 300 Km

c)- COORDENADAS DAS PRAÇAS-SEDES (AUDITORES):

NUM	X	Y	NUM	X	Y
1	3850	1877	2	2793	2189
3	3850	1877	4	2563	2748
5	3939	853	6	4297	1199
7	4323	1229	8	2607	2887
9	3939	853	10	2563	2748
11	3939	853	12	2794	3051
13	2196	3773	14	3515	2692
15	4312	1333	16	2424	2952
17	3850	1877	18	2411	2680
19	2487	3260	20	4037	1649
21	2196	3773	22	2764	2980
23	2068	2265	24	4199	1511
25	4323	1229	26	2480	2903
27	2732	2629	28	2670	2856
29	2196	3773	30	4312	1333
31	3286	2782	32	3170	2977
33	2518	3434	34	2927	2982
35	2196	3773	36	3170	2853
37	2020	2706	38	2513	3501

39	2513	3501	40	2196	3773
41	2517	3425	42	2051	3966
43	2793	2189	44	4235	1349
45	2639	2288	46	3119	2653
47	3850	1877	48	4323	1229
49	2196	3773	50	2196	3773
51	2020	2706	52	3850	1877
53	4199	1511	54	3043	2991
55	2570	2914	56	2196	3773
57	4312	1333	58	2563	2748
59	4199	1511	60	4296	1083
61	3459	1005	62	2336	3024
63	3939	853	64	3922	872
65	3939	853	66	4210	1242
67	4210	1242	68	4304	1278
69	3303	722	70	2720	2788
71	2764	2980	72	2196	3773
73	1917	4161	74	2196	3773
75	2020	2706	76	2639	2288
77	2732	2629	78	2707	2959
79	2607	2887	80	2639	2288
81	2793	2189	82	2336	3024
83	3515	2692	84	3170	2977

d)-COORDENADAS E TEMPOS DE AUDITORIA DAS AGÊNCIAS

NUM	X	Y	T	NUM	X	Y	T
85	3769	1962	14	86	3771	1985	14
87	2987	2204	14	88	3119	2207	14
89	2785	2241	20	90	2580	2165	20
91	3886	1664	14	92	3736	1625	14
93	2638	2758	14	94	3734	1039	14
95	3954	1307	14	96	2689	3008	14
97	2532	3086	14	98	2794	3051	20
99	2588	2803	14	100	3929	883	14
101	1432	241	14	102	2559	2773	14
103	2809	2875	20	104	3667	855	14
105	3886	819	14	106	2500	3069	14
107	2607	2887	20	108	2442	3615	14
109	2058	3616	14	110	3497	2700	14
111	3505	2476	14	112	3400	2708	14
113	4011	1300	14	114	4292	1359	20
115	3886	1312	14	116	2481	2688	20
117	3684	1939	20	118	3685	1994	14
119	2649	3010	14	120	2651	3155	20
121	2425	2703	14	122	2341	3281	14
123	2057	3314	14	124	3974	1675	14
125	4032	1643	14	126	1837	3798	20
127	2072	3471	14	128	2076	3463	14

129	2163	2252	14	130	2209	2359	20
131	4145	1562	20	132	4033	1472	14
133	4065	1482	14	134	4238	1234	14
135	4315	1265	14	136	3921	1230	14
137	4273	1223	14	138	2614	3044	14
139	3290	2610	20	140	3148	2637	27
141	3158	2615	14	142	2529	2743	14
143	2726	2721	14	144	2272	3707	20
145	4312	1333	27	146	4007	1337	14
147	4141	1452	14	148	4139	1424	14
149	2709	2286	14	150	3177	2892	20
151	3094	2949	14	152	3249	2976	20
153	2210	3730	20	154	2735	3012	14
155	2693	3021	14	156	2829	3034	14
157	2675	2754	14	158	2941	2906	14
159	3230	2803	14	160	2954	2778	20
161	1746	2221	20	162	1721	2547	20
163	2487	3260	20	164	2131	3431	14
165	2067	3442	14	166	2082	3443	14
167	2292	3756	20	168	2210	3742	14
169	2034	3586	14	170	2490	3466	14
171	2482	3467	14	172	1870	3530	14
173	1978	3497	14	174	1370	1708	20
175	1321	1679	14	176	1318	1646	20
177	1286	1628	14	178	2638	2224	14
179	2741	2404	14	180	4312	1333	14
181	4103	1435	14	182	4142	1438	14
183	4054	1450	14	184	2277	2387	20
185	2727	2222	14	186	2618	2253	20
187	2967	2635	14	188	3159	2979	14
189	3769	1877	20	190	3778	1940	14
191	3835	1830	14	192	4158	1223	14
193	4297	1199	14	194	4280	1173	14
195	4162	1271	14	196	2211	3768	14
197	1968	3787	20	198	2270	3749	20
199	2189	3650	20	200	3350	1790	20
201	1887	2783	14	202	2396	2666	14
203	3929	1483	20	204	3701	1600	20
205	4210	1428	14	206	4080	1544	14
207	2627	2977	20	208	2500	2979	20
209	2774	3048	14	210	2826	3010	14
211	2744	3000	14	212	2357	3695	20
213	2251	3698	14	214	4136	1406	14
215	3900	1394	14	216	2673	3030	20
217	2680	2769	14	218	2613	2739	20
219	2757	2963	14	220	4000	1452	14
221	4147	1479	14	222	4266	1159	14
223	3577	915	14	224	3596	1031	14
225	3424	1191	20	226	2371	3047	14
227	2426	3084	20	228	4148	1135	20
229	3686	763	14	230	3836	1193	14
231	3864	1173	14	232	3850	1276	14
233	3889	1260	14	234	3679	890	14
235	3677	876	14	236	3850	924	14
237	3946	904	14	238	3959	1250	14
239	3984	1218	14	240	3917	1204	20
241	4121	1270	14	242	4098	1202	14
243	4231	1168	14	244	4273	1311	14

245	4148	1426	14	246	4252	1189	14
247	3279	889	14	248	3194	889	14
249	2520	3012	20	250	2589	3079	14
251	2888	2778	14	252	2856	2882	14
253	2710	3014	14	254	2303	3767	14
255	2235	3654	14	256	2010	3729	14
257	2215	3778	20	258	2202	3892	14
259	2196	3773	14	260	2056	3456	14
261	2100	3475	20	262	2392	2621	20
263	2725	1356	20	264	3086	1450	14
265	2682	2647	14	266	2846	2928	14
267	2774	2518	14	268	2769	3070	14
269	2841	3025	14	270	2738	2695	20
271	2732	2704	14	272	2727	2770	14
273	2744	2878	20	274	2793	2189	20
275	2793	2189	20	276	2487	2485	20
277	2918	627	14	278	1032	813	14
279	3421	2724	14	280	3381	2742	20
281	2798	3067	20	282	2776	3011	20
283	2794	3051	20	284	1482	131	20
285	2720	2788	20	286	2101	3294	20
287	2710	2537	20	288	2806	3041	20
289	2424	2952	20	290	2037	3535	20
291	2793	2189	20	292	2203	3760	20
293	1766	3868	20	294	2226	3729	20
295	1948	3521	20	296	3159	2979	20
297	3159	2979	20	298	2333	2893	20
299	3939	853	20	300	2794	3051	20
301	2816	3097	20	302	3459	1005	20
303	3159	2979	20				

e)-IMPEDIMENTOS PARA OS AUDITORES:

AUDITOR	AGENCIAS
8	107
16	289
42	295
71	282
72	153
79	154

f)-SOLUÇÃO GERADA PELO SISROTAU

AUDITOR	ROTA					TR	DIST
1	191	0	0	0	0	14	98
2	291	88	87	0	0	42	653
3	190	85	86	117	0	62	418
4	121	202	262	116	0	68	450
5	299	100	0	0	0	28	64
6	246	243	222	194	193	70	163
7	134	137	0	0	0	28	172
8	250	120	138	0	0	48	565
9	231	230	235	104	229	70	1104
10	270	157	217	93	0	62	404
11	299	0	0	0	0	14	0
12	283	300	268	98	0	62	62
13	292	294	153	168	0	62	116
14	111	112	279	0	0	42	596
15	195	244	0	0	0	28	325
16	227	106	97	249	0	68	432
17	92	204	200	118	0	68	1313
18	130	129	184	0	0	54	994
19	286	122	163	0	0	48	775
20	124	91	125	0	0	42	313
21	254	167	198	0	0	54	224
22	283	300	209	0	0	42	166
23	175	176	177	278	0	62	3657
24	147	181	183	220	132	70	440
25	135	0	0	0	0	14	74
26	289	298	208	0	0	48	449
27	287	265	0	0	0	28	262
28	282	285	160	103	0	68	966
29	292	294	128	260	127	70	728
30	114	0	0	0	0	20	66
31	303	159	139	280	0	68	889
32	296	297	152	0	0	48	180
33	273	219	211	0	0	48	1214
34	301	266	252	251	158	70	704
35	255	212	144	213	0	68	456
36	303	150	0	0	0	34	255
37	174	161	0	0	0	40	2384
38	108	0	0	0	0	14	268
39	171	170	0	0	0	28	96
40	164	166	165	261	0	62	773
41	293	126	197	256	0	68	845
42	185	89	0	0	0	34	188
43	214	148	182	245	205	70	304
44	149	0	0	0	0	14	140
45	187	141	140	0	0	55	402
46	189	0	0	0	0	20	162
47	290	295	172	169	109	70	875
48	290	295	173	199	0	62	798
49	162	201	0	0	0	34	782
50	146	215	203	0	0	48	745
51	301	0	0	0	0	14	502

52	155	216	119	207	0	68	341
53	258	0	0	0	0	14	238
54	145	180	0	0	0	41	0
55	142	102	99	218	0	62	238
56	131	206	133	221	0	62	348
57	284	101	263	0	0	48	4804
58	289	298	226	0	0	42	423
59	302	264	225	224	0	62	2131
60	302	248	247	223	234	70	1504
61	237	94	236	105	0	56	639
62	233	232	115	95	113	70	747
63	241	242	228	192	0	62	393
64	238	136	240	239	0	62	809
65	284	277	0	0	0	28	3831
66	282	285	143	271	272	70	630
67	281	288	156	269	210	70	249
68	257	196	0	0	0	34	47
69	293	0	0	0	0	14	660
70	259	0	0	0	0	14	0
71	179	276	0	0	0	34	670
72	287	267	0	0	0	28	281
73	281	288	154	253	96	70	344
74	107	0	0	0	0	20	0
75	186	90	178	0	0	54	284
76	291	274	275	0	0	54	0
77	286	123	0	0	0	28	808
78	110	0	0	0	0	14	40
79	296	297	188	151	0	56	164
80	0	0	0	0	0	0	0
81	0	0	0	0	0	0	0
82	0	0	0	0	0	0	0
83	0	0	0	0	0	0	0
84	0	0	0	0	0	0	0

OBS: 1)- Reservamos cinco posições para os nós nas rotas de cada auditor. Da esquerda para a direita os nós aparecem na sequência em que foram roteados. Quando a posição não contém nenhum nó, colocamos '0'.

2)- Sob a coluna 'TR' listamos os tempos de cada cada rota e sob a coluna 'DIST', as distâncias de cada rota.

3)- Os nós {281,282,...,303} correspondem a agências do tipo missão conjunta.

g)- MISSÕES CONJUNTAS FORMADAS

AUDITORES	MISSÃO CONJUNTA	TIPO
2 e 81	291	I
5 e 11	299	I
12 e 22	283 e 300	II
13 e 29	292 e 294	II
19 e 82	286	I
26 e 62	289 e 298	II
27 e 77	287	I
28 e 70	282 e 285	II
31 e 36	303	I
32 e 84	296 e 297	II
34 e 54	301	I
42 e 73	293	I
49 e 50	290 e 295	II
61 e 69	284	I
63 e 64	302	I
71 e 78	281 e 288	II

II - Problema Real Dois (PR2)

a)-DIMENSÕES:
- número de auditores (M) = 77
- número de agências do tipo missão simples (NAS) = 164
- número de agências do tipo missão conjunta = 32

b)- PARÂMETROS:
- LAMBDA = 1.5
- MI = 2.5
- TMAXROT = 72 dias
- TMAXREFIN = 15 min
- DISTMAX = 1500 Km
- RMAX = 300 Km

c)- COORDENADAS DAS PRAÇAS-SEDES (AUDITORES):

NUM	X	Y	NUM	X	Y
1	2196	3773	2	4312	1333
3	4296	1083	4	2639	2288
5	3520	2693	6	2210	3742
7	4323	1229	8	3303	722
9	3246	2811	10	2447	3023
11	3688	1226	12	3282	2977
13	2764	2980	14	2710	2537
15	2639	2288	16	2710	2537
17	1920	2168	18	2487	3260
19	2765	3122	20	3939	853
21	2764	2980	22	2336	3024
23	2861	2820	24	3042	2931
25	2570	2914	26	2837	603
27	3170	2977	28	2426	2872
29	2513	3501	30	3170	2977
31	3637	2383	32	4312	1333
33	4312	1333	34	2487	3260
35	3850	1877	36	2563	2748
37	3811	1799	38	3148	2637
39	2720	2788	40	2837	2855
41	2696	2881	42	3850	1877
43	2732	2629	44	2487	3260
45	3850	1877	46	3170	2853
47	2720	2788	48	3738	2079
49	2513	3501	50	3850	1877
51	2513	3501	52	3939	853
53	2196	3773	54	1954	3734
55	3850	1877	56	3378	2529
57	2220	3676	58	2726	2868
59	3850	1877	60	3170	2853

61	2210	3730	62	3520	2693
63	4296	1083	64	2710	2537
65	3811	1799	66	2563	2748
67	4312	1333	68	2336	3024
69	3159	2979	70	3148	2637
71	4312	1333	72	2563	2748
73	3303	722	74	2673	3030
75	2290	3349	76	2513	3501
77	1954	3734			

d)-COORDENADAS E TEMPOS DE AUDITORIA DAS AGÊNCIAS

NUM	X	Y	T	NUM	X	Y	T
78	2182	3706	14	79	2121	3497	14
80	2208	3520	20	81	4181	1516	14
82	3697	1280	20	83	3735	1289	14
84	4143	1069	14	85	4094	1110	14
86	2583	1911	14	87	2984	2002	20
88	2718	1740	20	89	3448	2769	20
90	2000	915	20	91	4302	1195	14
92	4002	1191	14	93	3251	982	14
94	3320	2725	14	95	3299	2668	14
96	2461	3008	20	97	2481	3037	14
98	2282	3057	20	99	3568	1151	14
100	3424	1190	14	101	2783	2961	14
102	2801	3059	14	103	2908	2816	20
104	2791	1940	14	105	2937	1873	14
106	2609	2157	14	107	2363	2553	14
108	2951	2406	20	109	2928	2829	14
110	2037	2209	20	111	2308	3101	14
112	2470	3278	14	113	2507	3114	20
114	2563	3126	14	115	3757	921	14
116	3734	1039	14	117	2405	3007	14
118	2313	3073	14	119	2257	3099	14
120	2368	3032	20	121	2461	3046	14
122	2447	3023	20	123	2415	2683	14
124	2346	2831	14	125	2328	2821	20
126	2876	586	14	127	2936	567	14
128	2807	651	14	129	3159	2979	20
130	2712	3051	14	131	2825	2945	14
132	2720	2788	20	133	2833	2996	14
134	2058	3443	14	135	2115	3425	14
136	2068	3362	14	137	2371	3649	14
138	2448	3598	20	139	2504	3490	14
140	2622	2912	20	141	3845	1335	20
142	3886	1312	14	143	4270	1418	14
144	4222	1328	14	145	2827	3050	20
146	3908	1556	14	147	3849	1655	14
148	2355	2895	20	149	2340	2904	14
150	3731	2263	14	151	3783	1922	20

152	3754	1855	14	153	3731	1884	14
154	2907	2918	14	155	3338	2633	20
156	2643	2813	20	157	2654	2762	20
158	2891	2796	14	159	2899	2765	14
160	2867	2804	14	161	2744	2900	14
162	2781	3014	14	163	3653	1754	20
164	3646	1740	14	165	3685	1765	14
166	3524	2261	14	167	3517	2286	14
168	3575	2255	14	169	2338	2990	20
170	2348	3020	14	171	3758	2177	14
172	3700	2007	14	173	2917	2907	20
174	2990	2922	14	175	2290	2865	20
176	2378	2848	14	177	2807	2834	20
178	2318	3350	14	179	2518	3434	14
180	2011	3454	20	181	3859	1827	14
182	3874	1809	14	183	3669	1855	14
184	3540	1881	14	185	2043	3446	14
186	2117	3404	14	187	1857	3536	14
188	2213	3723	14	189	2119	3575	14
190	2031	3695	14	191	2058	3552	14
192	1892	3563	20	193	3885	1746	14
194	3911	1716	14	195	3012	2709	14
196	2802	2860	14	197	3095	2655	14
198	3738	2010	14	199	3652	2278	14
200	3738	2244	14	201	3155	2685	14
202	2847	2910	14	203	2861	2963	14
204	1963	3666	20	205	3523	2671	20
206	3305	2874	14	207	3339	2840	14
208	4109	1059	20	209	4010	1069	14
210	4143	1008	20	211	2618	2500	14
212	3119	2653	20	213	3905	1593	14
214	2450	2687	20	215	2338	2825	14
216	4112	1408	14	217	2794	3051	27
218	2813	3069	14	219	3139	2803	14
220	3329	2773	14	221	3171	2935	20
222	3282	2977	20	223	3116	2673	14
224	3215	2463	20	225	3228	2615	20
226	4115	1441	14	227	2503	3082	14
228	2812	3003	20	229	2604	2852	20
230	3337	913	14	231	3455	1006	14
232	3424	1071	14	233	2737	2984	14
234	2844	2982	14	235	2355	2830	14
236	2449	2668	14	237	2495	3245	14
238	2487	3260	20	239	2256	3446	14
240	1905	3527	20	241	1992	3510	14
242	3286	2782	20	243	3340	2685	20
244	3159	2979	20	245	3076	2937	20
246	2328	2201	20	247	2199	3106	20
248	2806	3041	20	249	2801	3059	20
250	2696	2881	20	251	2255	3037	20
252	2778	3049	20	253	3214	2295	20
254	2736	2942	20	255	2196	3773	20
256	3159	2979	20	257	3159	2979	20
258	3159	2979	20	259	3763	2079	20
260	2512	3357	20	261	2794	3051	20
262	2816	3097	20	263	2301	3051	20
264	2939	2926	20	265	2487	3260	20
266	2480	2903	20	267	3939	853	20

268	2187	3677	20	269	3148	2637	20
270	4296	1083	20	271	2798	3067	20
272	2794	3051	20	273	2517	3425	20

e)-IMPEDIMENTOS PARA OS AUDITORES:

AUDITOR	AGENCIAS
10	122
14	211
18	170
25	235
34	238
35	152

f)- SOLUÇÃO GERADA PELO SISROTAU

AUDITOR	ROTA					TR	DIST
1	255	268	118	111	0	56	1422
2	143	81	226	216	144	70	586
3	270	84	208	210	0	68	420
4	246	86	88	104	0	62	1520
5	155	225	89	0	0	60	677
6	188	0	0	0	0	14	38
7	92	209	85	0	0	42	796
8	93	100	232	231	230	70	1071
9	242	95	94	220	207	70	439
10	247	119	98	122	0	68	537
11	142	141	83	82	0	68	476
12	257	258	222	0	0	48	246
13	261	272	228	101	0	62	206
14	107	0	0	0	0	14	694
15	246	106	0	0	0	28	741
16	105	87	108	0	0	54	1518
17	110	0	0	0	0	20	248
18	260	265	238	0	0	48	200
19	261	272	217	102	0	69	161
20	267	115	0	0	0	28	388
21	248	252	218	145	0	62	261
22	251	263	120	117	0	62	316
23	249	271	133	131	202	70	518
24	245	264	154	173	0	62	347
25	262	140	0	0	0	34	627
26	90	128	127	126	0	62	2002
27	244	256	129	0	0	48	22
28	266	148	149	175	0	68	403
29	112	178	0	0	0	28	642

30	257	258	221	206	0	62	374
31	253	167	166	168	199	70	998
32	91	0	0	0	0	14	276
33	227	97	96	121	0	62	518
34	182	181	0	0	0	28	146
35	214	236	123	125	0	68	593
36	184	164	163	165	0	62	640
37	243	269	197	195	223	70	709
38	262	196	0	0	0	28	670
39	249	271	177	160	0	62	575
40	250	254	233	130	114	70	633
41	153	183	152	0	0	42	370
42	132	156	157	0	0	60	446
43	260	265	237	113	0	62	496
44	259	198	172	151	0	62	531
45	242	219	0	0	0	28	343
46	159	109	103	158	0	62	471
47	259	171	200	150	0	56	397
48	273	179	0	0	0	28	152
49	138	139	0	0	0	34	253
50	267	0	0	0	0	14	0
51	255	268	78	0	0	42	193
52	241	240	187	192	0	68	591
53	253	224	0	0	0	34	630
54	180	185	134	79	0	62	641
55	250	254	161	0	0	42	186
56	245	264	0	0	0	28	505
57	189	80	137	0	0	48	673
58	205	0	0	0	0	20	44
59	270	0	0	0	0	14	0
60	211	0	0	0	0	14	198
61	147	146	213	194	193	70	555
62	266	229	0	0	0	34	422
63	251	263	169	170	0	62	246
64	244	256	234	203	174	70	653
65	243	269	201	212	0	62	526
66	215	124	235	176	0	56	496
67	116	99	0	0	0	28	1239
68	248	252	162	0	0	42	306
69	247	136	186	135	239	70	879
70	273	0	0	0	0	14	152
71	190	191	204	0	0	48	449
72	0	0	0	0	0	0	0
73	0	0	0	0	0	0	0
74	0	0	0	0	0	0	0
75	0	0	0	0	0	0	0
76	0	0	0	0	0	0	0
77	0	0	0	0	0	0	0

OBS: Os nós {242,243,...,273} correspondem às agências do tipo missão conjunta.

g)- MISSÕES CONJUNTAS FORMADAS

AUDITORES	MISSÃO CONJUNTA	TIPO
1 e 53	255 e 268	II
3 e 63	270	I
4 e 15	246	I
9 e 46	242	I
10 e 75	247	I
12 e 30	257 e 258	II
13 e 19	261 e 272	II
18 e 44	260 e 265	II
20 e 52	267	I
21 e 74	248 e 252	II
22 e 68	251 e 263	II
23 e 40	249 e 271	II
24 e 60	245 e 264	II
25 e 39	262	I
27 e 69	244 e 256	II
28 e 66	266	I
31 e 56	253	I
38 e 70	243 e 269	II
41 e 58	250 e 254	II
49 e 76	273	I