# *HEURÍSTICAS PARA O PROBLEMA DO CAIXEIRO-VIAJANTE DE GRANDE PORTE*

*Paulo Victor Xavier Djmal*

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

Aprovada por:

Prof. Paulo Roberto de Oliveira, Dr. Ing.
(presidente)

Prof. Ruy Eduardo Campello, D. Sc.

Profa. Nair Maria Maia de Abreu, D. Sc.

Profa. Lilian Markenzon, D. Sc.

RIO DE JANEIRO, RJ - BRASIL
NOVEMBRO DE 1994

DJMAL, PAULO VICTOR XAVIER

Heurísticas para o Problema do Caixeiro-Viajante de Grande Porte

[ Rio de Janeiro ] 1994

VII, 79p., 29.7 cm (COPPE/UFRJ, M.Sc., ENGENHARIA DE SISTEMAS E COMPUTAÇÃO, 1994)

TESE - Universidade Federal do Rio de Janeiro, COPPE

1 - Otimização Combinatória 2 - Problema do Caixeiro-Viajante

3 - Heurística

I. COPPE/UFRJ II. Título(Série).

AGRADECIMENTO

A professora Nair Maria Maia de Abreu cujo inestimável apoio e colaboração foram de fundamental importância na elaboração deste trabalho.

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

Heurísticas Para o Problema do Caixeiro-Viajante de Grande Porte

Paulo Victor Xavier Djmal

Novembro de 1994

Orientadores : Ruy Eduardo Campello

Paulo Roberto de Oliveira

Programa: Engenharia de Sistemas e Computação

O Problema do Caixeiro-Viajante (PCV) é um problema clássico da otimização combinatória que tem merecido grande atenção por parte dos pesquisadores nos últimos trinta anos. Como trata-se de um problema para o qual é improvável a existência de algoritmo que obtenha a solução ótima em tempo polinomial, a busca de bons algoritmos heurísticos tem sido o enfoque principal nos últimos anos. Neste trabalho são descritos métodos já utilizados para tratar instâncias de grande porte bem como são propostas duas novas heurísticas baseadas na redução do problema por técnicas de grupamento. Os primeiros testes indicaram que estas heurísticas podem obter soluções de boa qualidade para instâncias geométricas de até 10.000 nós em estações de trabalho RISC 6000.

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

Heuristics for Large-Scale Travelling Salesman Problem

Paulo Victor Xavier Djmal

November, 1994

Thesis Supervisors: Ruy Eduardo Campello

Paulo Roberto de Oliveira

Department: Programa de Engenharia de Sistemas e Computação

The Travelling Salesman Problem (TSP) is a classical combinatorial optimization problem that has deserved much attention from researchers in the last thirty years. Considering that it is improbable that there is an alghorithm that can get to the optimal solution in polinomial time for TSP, the search for good heuristics has been the main approach adopted lately. In this work, we describe some methods that have already been used to deal with large-scale TSP and also propose two new heuristics based on problem reduction by means of clustering techniques. The first tests have indicated that this heuristics can obtain good solutions for geometric instances of up to 10.000 nodes in RISC 6000 workstations.

# Índice

# CAPÍTULO I

## Introdução

## *I.1 - Os Problemas de Otimização*

Um grande número de problemas teóricos e/ou práticos dizem respeito à escolha de uma determinada configuração ou conjunto de parâmetros para atingir certo objetivo. Problemas com estas características são denominados *problemas de otimização* e podem ser naturalmente divididos em dois grupos: os de *variáveis contínuas* e os de *variáveis discretas,* que também são conhecidos como *combinatórios* [*Papadimitriou e Steiglitz*, 27] . Nos problemas contínuos, geralmente procuramos um elemento, um conjunto ou mesmo uma função num universo não enumerável. No caso dos problemas combinatórios, o objetivo é encontrar um objeto, normalmente um inteiro, conjunto, permutação ou grafo, que pertence a um conjunto finito ou enumerável. Quando nos referimos a um determinado *problema de otimização,* estamos aí incluindo todas as *instâncias* possíveis deste problema, que normalmente formam um conjunto não finito. Uma instância é um conjunto de dados que se encaixa na definição do problema, ou mais precisamente, um par $(F,c)$ onde $F$ é o conjunto de todas as possíveis soluções e $c$ uma função relacionada a custo, tal que $c: F \rightarrow \mathbb{R}$ . A *solução ótima* da instância é $f \in F$ tal que $c(f) \leq c(y)$ para todo $y \in F$ . Admite-se que um algoritmo soluciona um determinado problema quando ele chega à solução ótima independente da instância a que tenha sido aplicado. Os métodos utilizados para solucionar problemas contínuos e combinatórios são, em geral, completamente diferentes e, neste trabalho, nosso estudo será dirigido a um dos problemas clássicos do segundo grupo.

## *I.2 - O Problema do Caixeiro-Viajante (PCV)*

Consideremos a seguinte situação: um caixeiro-viajante deve visitar uma determinada lista de cidades uma única vez e retornar ao ponto de partida. Assumindo que ele conheça as distâncias entre cada par de cidades, ele certamente dispõe de todas as informações necessárias para calcular o itinerário mais curto, embora não seja conhecida nenhuma maneira óbvia de fazê-lo. A escolha deste caminho mais curto, denominado *caminho* ou *tour ótimo* é conhecida na literatura como Problema do Caixeiro-Viajante (PCV), não se tendo obtido até hoje nenhum algoritmo que o solucione em tempo polinomial [*Garey e Johnson*, 30] .

Na verdade, não existem muitos caixeiros-viajantes necessitando de um algoritmo eficiente e as aplicacões comerciais e/ou industriais em que o uso do modelo matemático do PCV é adequado não são muito numerosas (embora haja uma grande variedade de problemas particulares que podem ser adaptados ao modelo, como pode ser visto em *Melamed et alii* [28] ). Por que então o PCV é considerado tão importante? A resposta é simples. Sua importância advém do fato de ser um típico problema de otimização combinatória, sendo extremamente simples de formular e, no entanto, intratável formalmente [*Hoffman e Wolfe*, 31] . Em 1985, *Lawler et alii* [41] editaram o livro *"The traveling salesman problem: a guided tour of combinatorial optimization"* que faz uma abordagem completa da otimização combinatória através da análise do PCV. Por tudo isso, o PCV é o mais célebre dos problemas de otimização combinatória que permanecem até hoje sem solução, sendo utilizado frequentemente como termo de comparação com outros problemas do gênero. A procura de algoritmos eficientes para solucionar problemas como este influenciou o desenvolvimento de diversas

áreas da ciência da computação, entre as quais a teoria da complexidade e a análise probabilística, empírica e garantia de performance de algoritmos heurísticos.

## *I.3 - Considerações sobre Teoria dos Grafos*

Para que mais tarde possamos apresentar a definição formal do PCV, bem como os diversos enfoques de tratamento do problema, é necessário que antes sejam introduzidos alguns conceitos sobre teoria dos grafos. Um grafo $G(V,E)$ é definido por um conjunto $V$ de *vértices*, alguns dos quais ligados por *arestas* (que constituem o conjunto $E$, cujos elementos são representados por pares de vértices, ordenados ou não). Dois vértices são ditos *adjacentes* quando são ligados por uma aresta. O *grau* de um vértice é definido como o número de vértices adjacentes a ele. Um grafo é dito *completo* quando existe uma aresta entre cada par de seus vértices. A notação $K_n$ é então usada para designar um grafo completo(também chamado *clique* ) com $n$ vértices. Um *caminho* é uma sequência de vértices $v_1, ..., v_k$ tal que $(v_j, v_{j+1}) \in E$, $1 \leq j < |k-1|$ . Se todos os vértices do caminho forem distintos, ele é denominado *simples*. Um *ciclo* é um caminho $v_1,...,v_k, v_{k+1}$ onde $v_1 = v_{k+1}$ e $k \geq 3$ . Se o caminho $v_1,..., v_k$ for simples, o ciclo também é dito simples. Um caminho ou ciclo que contenha todas as arestas de um grafo exatamente uma vez é chamado *euleriano* enquanto que um que contenha todos os vértices exatamente uma vez é chamado *hamiltoniano*.

O PCV para um grafo em que o comprimento das arestas é definido se resume em encontrar o ciclo hamiltoniano de menor comprimento, enquanto que o problema de decidir se um determinado grafo $G$ tem ou não um ciclo hamiltoniano é um caso particular do PCV como se pode concluir do

seguinte argumento: se criarmos o grafo $H$ a partir do grafo $G$, atribuindo as suas arestas o comprimento zero e adicionarmos a ele todas as arestas que lhe faltam atribuindo-lhes o comprimento 1, bastará solucionar o PCV para $H$ para verificar se o grafo $G$ contém ou não um ciclo hamiltoniano, pois é fácil notar que o tour de comprimento mínimo em $H$ é zero se e somente se existe um ciclo hamiltoniano em $G$.

A literatura existente sobre teoria dos grafos é bastante vasta e maiores esclarecimentos podem ser obtidos em várias fontes tais como *Berge* [33, 34] , *Tutte* [29] , *Harary* [35] , *Golumbic* [36] , *Swarcfiter* [32] e *Campello* [49] .

## *I.4 - Definição Matemática e Características Principais*

Como já foi visto anteriormente, o PCV é um problema extremamente simples de formular em linguagem corrente. Apesar disso, é um problema tão rico que admite várias formulações matemáticas, que variam de acordo com o tratamento que se quer dar ao problema. Vamos aqui discutir apenas as mais comuns:

- *Formulação por teoria dos grafos:* Seja $G(V,E)$ um grafo completo com pesos $c_{ij}$ para cada aresta $(i,j) \in E$ . A solução para o PCV é obtida encontrando-se um circuito hamiltoniano $H$ tal que $\sum_{(i,j) \in H} c_{ij}$ seja mínimo [*Parker e Rardin*, 12] .

- *Formulação por programação linear inteira:* Seja $C = (c_{ij})$ uma matriz $n \times n$ onde cada elemento $c_{ij}$ representa a distância

entre as cidades $i$ e $j$. O modelo clássico proposto por *Dantzig, Fulkerson e Johnson* [01] é

$$\text{minimizar } \sum c_{ij}\, x_{ij}$$

$$\text{sujeito à: } \sum_i x_{ij} = 1 \qquad (1)$$

$$\sum_j x_{ij} = 1 \qquad (2)$$

$$x_{ij} \in \{0,1\} \qquad (3)$$

$$X^x = \{x_{ij}\} \in H \qquad (4)$$

A restrição (4) é necessária para restringir a solução aos ciclos hamiltonianos, evitando-se assim a possibilidade de subtours. Há na literatura diferentes propostas de expressar esta restrição. Uma boa fonte de consulta é *Bodin et alii*[62] .

- *Formulação combinatória:* Seja $A = [a_{ij}]$ uma matriz real quadrada $n \times n$ onde cada elemento $a_{ij}$ representa a distância entre as cidades $i$ e $j$. A solução para o PCV é uma permutação $(i_1, i_2, \ldots, i_n)$ dos inteiros $1,2,\ldots n$ tal que a soma $a_{i1\,i2} + a_{i2\,i3} + \ldots + a_{in\,i1}$ é mínima [*Karg e Thompson*, 15] .

Existem algumas características do problema que geralmente influenciam o grau de dificuldade de se obter soluções ótimas ou boas aproximações. Uma instância do PCV é dita *simétrica* se as distâncias $d_{ij} = d_{ji}$ para todo $i$ e todo $j$. Se, além disso, as distâncias $d_{ij}$ são calculadas no plano (onde é válida a desigualdade triangular: $d_{ik} \leq d_{ij} + d_{jk}$ para todo $i,j,k$), o problema é dito *euclidiano*. No PCV não simétrico, a solução é encontrada entre as $(n-1)!$ possíveis permutações e, para o problema simétrico, entre as $\frac{1}{2}(n-1)!$ permutações. É evidente que a enumeração total das permutações levaria à

solução ótima do problema. Tal procedimento porém é inviável computacionalmente pois os tempos requeridos cresceriam de tal forma com a instância do problema que seriam necessários milhões de anos para enumerar todas as permutações possíveis para instâncias da ordem de apenas algumas dezenas de cidades. Existem ainda algumas outras variações e/ou casos particulares do PCV muito interessantes, tais como: o PCV gargalo, onde o objetivo é minimizar a maior distância entre cidades utilizada no ciclo e não a soma das distâncias; o PCV dependente do tempo, no qual a distância entre as cidades é função do custo do deslocamento entre elas em determinado instante; o PCV com restrições de precedência onde procura-se o menor tour sujeito à restrições de precedência entre um ou mais pares de cidades; o PCV por grupamentos, em que o conjunto de cidades é dividido em grupos de tal forma que todas as cidades de um mesmo grupo devem ser visitadas em sequência e o PCV seletivo que difere do anterior apenas por ser necessário visitar apenas uma das cidades de cada grupo. Uma descrição sucinta de uma grande variedade destes problemas correlatos pode ser encontrada em *Melamed et alii* [28] que descreve o estado da arte do problema em 1988 incluindo referências a grande número de contribuições existentes na literatura russa que não foram consideradas por *Lawler et alii* [41] . Recentemente, *Burkard e Van Der Veen* [51] apresentaram um interessante estudo sobre o PCV algébrico, no qual o custo do ciclo é uma composição algébrica de elementos que formam um semigrupo comutativo. Uma rica coletânea de casos particulares resolvidos do PCV pode ser encontrada em *Gilmore et alii* [50] .

## *I.5 - A História do PCV*

Não se sabe com certeza quem trouxe o nome PCV ao meio científico. Segundo *Flood* [02] , ele teria sido apresentado em 1934 por Hassler Whitney, num seminário na Universidade de Princeton. Muitos anos antes porém, problemas semelhantes já haviam sido discutidos. No século XVIII, por exemplo, Euler e Vandermonde já discutiam o problema do *tour do cavalo* que consistia em encontrar o ciclo hamiltoniano para o grafo no qual o conjunto de vértices é representado pelas 64 casas do tabuleiro de xadrez, sendo dois vértices adjacentes se e somente se o movimento do cavalo é possível de uma casa para outra [*Biggs et alii*, 37] . No século XIX, Hamilton criou um sistema algébrico não comutativo ao qual deu o nome de *Icosian Calculus* (vértices adjacentes do dodecaedro correspondiam a faces adjacentes do icosaedro) e usou sua interpretação gráfica para criar um jogo, o *Icosian Game,* que consistia de vários problemas entre os quais o de completar um ciclo em que os primeiros passos eram fornecidos [*Hoffman e Wolfe*, 31] . Poucos anos antes do seminário de Whitney, Menger apresentou o *problema do mensageiro* que diferia do PCV apenas por procurar o menor caminho que ligasse um conjunto finito de pontos e não o menor ciclo [*Hoffman e Wolfe*, 31] . O artigo de *Dantzig, Fulkerson e Johnson* [01] publicado em 1954 é considerado um marco importante na história do PCV e da otimização combinatória não só pela proposta que apresenta (o uso da programação linear para chegar à solução ótima), mas por inspirar inúmeros desenvolvimentos futuros. Outras propostas de solução incluíam o uso de algoritmos *branch and bound* [*Eastman*, 40] e [*Croes*, 03] . Porém, a partir da década de 70, o rápido desenvolvimento da teoria da complexidade computacional reforçou em muito a conjectura de que a

existência de um algoritmo polinomial que solucione de forma ótima o PCV genericamente é extremamente improvável (embora haja instâncias particulares para as quais é possível obter tais algoritmos), tornando ainda maior a tendência dos pesquisadores em buscar algoritmos que garantam boas soluções aproximadas com tempo computacional não muito elevado. Vários métodos heurísticos foram propostos, alguns deles com boas garantias de performance como o de *Christofides* [42] que admite um erro de 50% no pior caso para o PCV euclidiano. A literatura sobre métodos heurísticos para o PCV é riquíssima, podendo-se consultar entre outros, os artigos de *Held e Karp* [6, 20] , *Golden et alii* [09] , *Karg e Thompson* [15] , *Raymond* [18] , *Rosenkrantz et alii* [21] , *Stewart* [22] , *Lin e Kernighan* [08] , que propõe um método que se revelou dos mais eficientes e que continua sendo largamente utilizado, e *Parker e Rardin* [23] , que descreve o estado da arte em 1983. Recentemente foram propostos métodos heurísticos analógicos tais como o de *Hopfield e Tank* [55] baseado em redes neuronais e o algoritmo elástico, desenvolvido por *Durbin e Willshaw* [56] . As instâncias de maior porte para as quais a solução ótima é conhecida foram apresentadas por *Lin e Kernighan* [08] com 318 cidades e por *Padberg e Rinaldi* [13, 24] com 532 e 2392.

## *I.6 - Aplicações Práticas do Modelo Matemático do PCV*

É possível resolver alguns problemas aparentemente não relacionados ao PCV transformando-os em instâncias do mesmo. Alguns exemplos são então mostrados a seguir:

- *Problema do roteamento de veículos:* dispondo-se de uma frota de veículos e de uma determinada lista de clientes a visitar, este problema consiste em determinar que clientes devem ser visitados por quais veículos e em que ordem as visitas devem ser feitas de forma a minimizar o percurso total dos veículos. Geralmente existem restrições quanto à capacidade dos veículos bem como sobre o período no qual cada cliente deve ser visitado. Uma visão geral deste problema é encontrada em *Christofides* [43] . *Lenstra e Rinnooy Kan* [11] estudaram duas instâncias práticas muito interessantes: na primeira delas, na província holandesa de North-Holland, a companhia telefônica deve visitar as cabines telefônicas de 28 cidades uma ou duas vezes por semana para recolher as fichas e fazer eventuais reparos. O dia de trabalho do pessoal técnico envolvido deve começar e terminar na capital da província e a companhia deseja minimizar o intervalo entre as visitas às cabines, bem como a distância total percorrida. Na segunda instância, na cidade de Utrecht, aproximadamente 200 caixas de correio devem ter seu conteúdo recolhido todos os dias dentro de um intervalo de uma hora por caminhões que operam a partir da estação central. O objetivo é encontrar o número mínimo de caminhões capaz de realizar a tarefa. Métodos heurísticos para problemas de roteamento podem ser encontrados em *Clarke e Wright* [44] , *Gillet e Miller* [45] , *Christofides et alii* [46] e *Fisher e Jaikumar* [47] .

- *Problema da ordenação de tarefas:* consideremos o problema de estabelecer a sequência de *n* tarefas em uma máquina de forma a executar a totalidade destas no menor tempo possível. Assumindo que a máquina deve estar num determinado estado $S_j$ (que pode estar relacionado com pressão, temperatura, rotação, etc.) para executar a tarefa *j*, e que o estado inicial da máquina é $S_0$ , então o tempo requerido para completar a tarefa *j* logo após a tarefa *i* é $t_{ij} = c_{ij} + p_j$ onde $c_{ij}$ é o tempo requerido para alterar o estado da máquina de $S_i$ para $S_j$ e $p_j$ é o tempo efetivamente gasto para executar a tarefa *j*. Para uma determinada permutação $\pi$ dos inteiros *0,1,...,n,* o tempo necessário para executar todas as tarefas será dado pela expressão: $\sum_{i=0}^{n}(c_{i\pi(i)} + p_{\pi(i)}) = \sum_{i=0}^{n} c_{i\pi(i)} + \sum_{j=0}^{n} p_j$ . Como $\sum_{j=0}^{n} p_j$ é constante, o problema da ordenação das tarefas cai diretamente na formulação do PCV, ou seja, minimizar $\sum_{i=0}^{n} c_{i\pi(i)}$ .

- *Problema da fiação impressa:* trata-se de um problema que ocorre com frequência no projeto de interfaces de computadores [*Lenstra e Rinnooy Kan*, 11] . Uma interface consiste de um certo número de módulos, em cada um dos quais se localizam vários pinos. A posição de cada módulo é pré-estabelecida e um determinado conjunto de pinos deve ser interligado. Em virtude das pequenas dimensões dos pinos, no máximo podem ser ligados dois fios a cada pino. O objetivo então é minimizar o comprimento total da fiação

necessária para ligar os pinos. Seja então $P$ o conjunto de pinos a ser interligado, $c_{ij}$ , a distância entre os pinos $i$ e $j$ e $H$ o grafo completo obtido a partir de $P$, com peso $c_{ij}$ em suas arestas. A restrição de só podermos ligar dois fios a cada pino caracteriza a solução como o menor caminho hamiltoniano existente em $H$. Isto é equivalente a encontrar o menor circuito hamiltoniano num grafo completo obtido a partir do conjunto de pinos $N = P \cup \{x\}$ , sendo $c_{ix} = c_{xi} = 0$ para todo $i \in N$ . Desta forma, o problema pode ser convertido em um PCV euclidiano simétrico. Um problema mais complexo seria admitir que as posições dos módulos pudessem ser escolhidas de forma a minimizar o comprimento total da fiação (ou seja, não seriam pré-fixadas). Este problema é um caso do *problema quadrático de alocação* (PQA), do qual o PCV é um caso particular. Uma outra abordagem interessante pode ser encontrada em *Reinelt* [38] que descreve tanto o problema da perfuração das placas de circuito quanto o da gravação da fiação. No caso da perfuração, devem ser feitos vários furos (normalmente o número varia entre algumas centenas e alguns milhares) de diversos diâmetros. A máquina pode movimentar-se nos eixos $x$ (movimento da placa) e $y$ (movimento da ferramenta) simultaneamente. Como o tempo para a troca de ferramenta tem ordem de grandeza bem superior ao do movimento da máquina, o problema na prática fica dividido em vários subproblemas cada qual composto pelos furos de mesmo diâmetro. Em virtude da composição

de movimentos da máquina tornar a distância real entre os pontos um problema muito complexo, Reinelt utilizou na modelagem a métrica de Chebyshev (valor máximo entre as distâncias nos eixos *x* e *y*), tendo obtido bons resultados. No caso da gravação dos fios, a máquina se utiliza de diversas lentes, que podem gerar diferentes estruturas na placa (pontos e linhas por exemplo). Como no caso anterior, o tempo requerido para troca de lentes é relativamente muito grande, e desta forma, uma boa solução deve fazer consecutivamente todas as exposições com uma mesma lente. Como o movimento da máquina também tem características muito semelhantes aos da máquina perfuradora, o autor modelou ambos os problemas de forma idêntica. Cabe ressaltar que as instâncias reais destes problemas normalmente são de grande porte e se constituem numa grande motivação para o estudo de heurísticas que a elas se possam aplicar com eficiência.

Várias outras aplicações poderiam ser citadas, mas não nos estenderemos no assunto, pois o objetivo aqui é apenas mostrar que existem problemas práticos aos quais a modelagem do PCV se enquadra perfeitamente e, desta forma, aumentar a motivação para o estudo do mesmo.

## *I.7 - Complexidade de Algoritmos*

Considera-se que um algoritmo soluciona um problema *P* se, dada qualquer instância *I* de *P* como entrada, o algoritmo chegará sempre a uma solução em tempo finito. Existem alguns problemas para os quais já foi provado que é impossível construir algoritmos que os solucionem. O PCV certamente não é um deles pois já foi visto anteriormente que a enumeração total das permutações nos levaria à solução de qualquer instância e tal procedimento, obviamente, pode ser feito em tempo finito. Porém, existe um verdadeiro abismo entre a teoria e a prática quando se trata de definir o que vem a ser *tempo finito,* pois, na prática, não se pode considerar milhares ou milhões de anos, por exemplo, como tempo finito. É necessário então redefinir o critério usado para se considerar um problema resolvido, passando a considerar satisfatoriamente resolvidos apenas os problemas para os quais se conhece pelo menos um algoritmo que aplicado a qualquer instância dos mesmos, chega à solução em tempo computacional aceitável. A dificuldade agora vai se concentrar em definir o que vem a ser tempo computacional aceitável. Até meados da década de 60, as medidas de eficiência de algoritmos eram apenas empíricas, ou seja, executava-se o algoritmo para determinadas instâncias arbitrárias e traçava-se sua curva de performance a partir dos resultados obtidos. É claro que tais medidas dependiam não só das instâncias consideradas mas também da qualidade do programador e do hardware e software utilizado. Era necessário adotar um critério analítico para avaliação de eficiência que evitasse estes inconvenientes. A *teoria da complexidade computacional,* desenvolvida principalmente no final da década de 60 e ao longo dos anos 70, veio então estabelecer um critério aplicável na maioria dos casos e que vamos agora

descrever de forma sucinta. Nos problemas de otimização combinatória, os tempos de execução dos algoritmos geralmente variam em função do tamanho das instâncias consideradas e de sua natureza. Como estamos interessados em soluções genéricas, devemos analisar o comportamento do algoritmo para a instância que necessite de maior tempo computacional, que se convencionou chamar de *pior caso*. Define-se então a *complexidade de pior caso* de um algoritmo como a função que associa o tempo de execução de um algoritmo com o tamanho da instância considerada no pior caso e, para que tal função seja independente da máquina utilizada, é admitida a simplificação de atribuir a cada passo do algoritmo o gasto de uma unidade de tempo, independente da instrução utilizada. Esta simplificação não introduz distorções significativas e permite que a complexidade de pior caso seja uma função de uma única variável: o tamanho da instância (um algoritmo pode ter complexidade de pior caso $10n^2$ ou $2^n$ por exemplo, onde $n$ é o tamanho da instância). A partir de agora, devemos concentrar a análise nos grandes valores de $n$, pois são eles que vão impor os limites de aplicabilidade do algoritmo. Para tais valores, a diferença entre $9n^3$ e $10n^3$ , por exemplo, pode ser considerada irrelevante, bem como podem ser considerados desprezíveis os termos com menor razão de crescimento, tais como $n$ em $n^3 + n$ . O objetivo então é obter valores aproximados que nos permitam ter uma idéia da ordem de grandeza do crescimento da complexidade do algoritmo, uma vez que, na prática, é difícil calcular o seu valor exato. A aproximação utilizada é a complexidade assintótica, representada pelo caracter $O$ e definida da seguinte forma: sejam $f(n)$ e $g(n)$ funções de inteiros positivos em reais positivos. Dizemos que $f(n)$ é $O(g(n))$ se, para qualquer valor de $n$ existe uma constante $c > 0$ tal que $f(n) \leq cg(n)$ ( então $3n^2 + n$ é $O(n^2)$ por exemplo). Com a estimativa fornecida pela complexidade

assintótica, foi então possível chegar a um consenso no que diz respeito à definição de tempo computacional aceitável. Existe hoje uma concordância geral em torno do critério proposto por *Edmonds* que considera um algoritmo satisfatório somente se sua complexidade for limitada por uma função *polinomial* do tamanho da entrada [*Papadimitriou e Steiglitz*, 27] . Assim, algoritmos de complexidade $O(n)$ e $O(n \log n)$ são considerados aceitáveis enquanto que um outro de complexidade $O(2^n)$ não o é. A tabela a seguir destaca a grande diferença entre a razão de crescimento das funções polinomiais e não polinomiais, as quais são genericamente chamadas de *exponenciais*.

| função | valores aproximados | | |
|---|---|---|---|
| | 10 | 100 | 1000 |
| $n$ | 10 | 100 | 1000 |
| $n \log n$ | 33 | 664 | 9966 |
| $n^3$ | 1000 | 1000000 | $10^9$ |
| $10^6 n^8$ | $10^{14}$ | $10^{22}$ | $10^{30}$ |
| $2^n$ | 1024 | $1.27 \times 10^{30}$ | $1.05 \times 10^{39}$ |
| $n^{\log n}$ | 2099 | $1.93 \times 10^{13}$ | $7.89 \times 10^{29}$ |
| $n!$ | 3628800 | $10^{158}$ | $4 \times 10^{2567}$ |

Podemos notar que existem funções polinomiais cujo comportamento dificilmente poderia ser considerado aceitável, como é o caso de $10^6 n^8$ , para a qual, em muitas instâncias, algoritmos exponenciais

poderiam ter performance superior. Este fato, entretanto, não contraria de forma alguma a tese que defende a qualidade dos algoritmos polinomiais, pois a experiência mostra que uma vez descoberto um algoritmo polinomial que solucione um problema, o grau do polinômio decresce rapidamente à medida que pesquisadores vão introduzindo aperfeiçoamentos no algoritmo e, normalmente, a complexidade final é $O(n^3)$ ou menor. A mesma coisa não ocorre com os algoritmos exponenciais, e estes geralmente são abandonados assim que se consegue um algoritmo polinomial para o problema. Maiores esclarecimentos sobre teoria da complexidade podem ser encontrados em *Papadimitriou e Steiglitz*[27] e em *Garey e Johnson*[30] , sendo esta última uma das referências mais recomendáveis sobre o assunto.

## *I.8 - A dificuldade do PCV*

Os problemas de decisão, cuja solução é caracterizada simplesmente por uma resposta afirmativa ou negativa, podem ser divididos em duas classes: a classe P composta por aqueles para os quais existe algoritmo polinomial para resolvê-los e a classe NP que compreende aqueles para os quais existe um algoritmo polinomial que, para qualquer instância, é capaz de verificar se uma resposta afirmativa está correta. Obviamente $P \subseteq NP$ e, embora existam fortes evidências de que $P \neq NP$ , esta ainda é uma questão em aberto. Se um problema é *indecidível,* ou seja não existe algoritmo que o resolva em tempo finito, ele certamente não pertence às classes P ou NP . Se, ao contrário, estivermos considerando um problema *decidível;* ou seja, um problema para o qual é possível construir um algoritmo que o resolva em tempo finito, não é possível provar formalmente que ele não pertence à classe P , ainda que se tenha fortes evidências neste sentido.

No início da década de 70, os artigos de *Cook*[53] e *Karp*[54] caracterizaram uma ampla classe de problemas em NP , a qual denominaram *NP-completos* que possuem as seguintes propriedades:

- não se sabe se algum deles pertence a P
- se algum deles pertencer a P então
  todos os outros também devem pertencer a P

O fato de um problema ser NP-completo é considerado como forte evidência da não existência de algoritmos polinomiais para a sua solução. Alguns exemplos de problemas NP-completos são o da *satisfabilidade* de fórmulas booleanas (SAT) e da determinação da existência de *k-colorações, k-cliques* e *ciclos hamiltonianos* em grafos. Os problemas de otimização correspondentes aos problemas de decisão NP-completos (caso do PCV, determinação do clique maximal, número cromático e etc..), são intuitivamente mais difíceis, pois quando a resposta é afirmativa no caso do problema de decisão, no problema de otimização estaremos ainda interessados na *melhor* solução. Assim, no caso do PCV, para uma resposta afirmativa ao problema de decisão (verificação da existência de ciclo hamiltoniano), basta apresentar um ciclo hamiltoniano qualquer; enquanto que para o problema de otimização, é necessário apresentar o *menor* ciclo. É claro que um algoritmo que resolvesse o problema de otimização também obteria uma resposta para o problema de decisão em tempo polinomial, o que provavelmente nos levaria a inferir que os problemas de otimização também pertenceriam à classe NP . Entretanto, até hoje não foi possível provar a pertinência destes problemas à NP , e, deste modo eles ficaram conhecidos na literatura como *NP-árduos* [*Papadimitriou e Steiglitz*, 27] . É interessante

notar que até hoje não foi provada a impossibilidade de se obter um algoritmo polinomial que resolva os problemas NP-completos; porém a conjectura é muito forte, não tendo sido até hoje contestada.

## *I.9 - Justificativa e Desenvolvimento da Tese*

Como já vimos, o PCV é um problema de otimização combinatória considerado intratável pela literatura. Desta forma, o esforço da pesquisa vem sendo feito no sentido de obter bons algoritmos heurísticos, e, neste aspecto, tal esforço pode ser considerado bem sucedido, tendo em vista a boa qualidade de vários algoritmos já desenvolvidos quando aplicados a instâncias de pequeno e médio porte. Entretanto, quando são aplicados a instâncias de grande porte, em muitos casos o tempo computacional requerido torna-se extremamente elevado, o que pode tornar impossível o seu uso em virtude de eventuais restrições de custo e/ou tempo. Desta forma, é interessante buscar heurísticas mais rápidas para as instâncias de grande porte (em particular as geométricas que caracterizam a maioria dos problemas práticos como já vimos na seção I.6) que possam satisfazer tais restrições sem, ao mesmo tempo, sacrificar em demasia a qualidade da solução obtida. E é exatamente neste contexto que vai se inserir nosso trabalho. O capítulo II apresenta uma discussão sobre o PCV de grande porte, mostrando suas características e técnicas de construção de heurísticas já propostas. Em seguida, no capítulo III, são propostas duas heurísticas que apresentam algumas variações e simplificações em relação àquelas já vistas no capítulo II, com o objetivo de torná-las ainda mais rápidas. No capítulo IV apresentamos uma análise dos resultados obtidos na aplicação destas heurísticas a problemas gerados aleatoriamente e finalmente no capítulo V,

são apresentadas as conclusões finais sobre o trabalho e perspectivas de continuar a pesquisa de algoritmos para instâncias de grande porte nesta direção.

# CAPÍTULO II

## O PCV de grande porte

## *II.1 - O significado*

Quando utilizamos o termo *PCV de grande porte,* não explicitamos claramente a que instância(s) estamos nos referindo. Cabe então a pergunta: o que vem a ser PCV de grande porte? A resposta não poderia ser outra: depende. Há uma série de fatores que devem ser considerados quando buscamos definir o termo e, dentre eles, os mais importantes são a qualidade da solução que se deseja obter, o grau de desenvolvimento da pesquisa em relação ao problema e com que tipo de ferramentas auxiliares se pode contar. Para maior clareza, vamos discutir cada um destes fatores isoladamente:

- *Qualidade da solução:* Se o objetivo é a solução ótima, talvez uma instância de menos de 100 nós já possa ser considerada de grande porte. Por outro lado, se a rapidez e/ou baixo custo forem predominantes em relação à qualidade, certamente o limite inferior cresceria para alguns milhares de nós. Para um caso intermediário, em que maior rapidez e menor custo tenham o mesmo peso que a qualidade da solução, provavelmente poderíamos estimar o limite em torno de 1000 nós.

- *Grau de desenvolvimento da pesquisa:* A análise feita no item anterior foi elaborada com base no estágio atual de desenvolvimento da pesquisa. No primeiro caso, a solução poderia ser obtida por programação linear ou métodos branch and bound; no segundo, por heurísticas simples (um algoritmo de vizinho mais próximo adaptado, por exemplo)

e, no terceiro, por heurísticas elaboradas que apresentam melhor performance, como as de *Christofides* [42] ou *Lin-Kernighan* [08] . É claro que esta análise seria bem diferente 40 anos atrás, quando a pesquisa ainda engatinhava no PCV, e instâncias de poucas dezenas de nós eram consideradas grandes. Da mesma forma, é possível que um desenvolvimento rápido da pesquisa possa tornar esta análise obsoleta nos próximos anos ou que, por outro lado, um desenvolvimento pouco acelerado (hipótese mais provável), a mantenha atualizada por um longo período.

- *Ferramentas auxiliares:* Normalmente o uso de ferramentas auxiliares está intimamente relacionado a *custo*. Dependendo da disponibilidade financeira e da relação custo/benefício, podemos ter como ferramenta auxiliar um supercomputador de última geração, um simples micro ou até mesmo lápis e papel somente. Para o último caso, até mesmo uma instância de pouco mais de uma dezena de nós é grande. As limitações dos microcomputadores, se comparadas às dos mainframes são bastante grandes e, por isso, os computadores de menor porte tendem a chegar bem antes a um limite máximo de nós além do qual a execução do algoritmo utlizado falhará em virtude daquelas limitações. Entretanto, também devemos levar em conta que este *gap* entre micros e mainframes, no que diz respeito a CPU e memória (requisitos fundamentais para o PCV) vem progressivamente sendo reduzido e, é bem possível que num horizonte não

muito distante este deixe de ser um fator de importância significativa.

A partir do que já foi comentado anteriormente, vamos definir que instâncias vamos considerar de grande porte neste trabalho. Nosso objetivo será concentrado em estudar a maioria dos casos, ou seja, aqueles em que não há predominância dos fatores custo e/ou tempo sobre a qualidade e vice-versa. Neste caso, normalmente o interesse é obter boas soluções sub-ótimas no menor tempo possível e a um custo relativamente baixo, o que sugere o uso de computadores de pequeno e médio porte como ferramentas auxiliares. Levando tudo isso em consideração, estabeleceremos como objeto de nosso estudo as instâncias a partir de 100 nós. O limite superior ficará em aberto, sendo determinado em função das limitações do hardware e software utilizados nos testes.

## *II.2 - As dificuldades*

Quando tratamos de instâncias de grande porte do PCV, enfrentamos alguns problemas que muitas vezes são de pequena ou nenhuma importância para instâncias menores. Por exemplo, quase todas as heurísticas convencionais obrigam o programador a calcular um vetor de distâncias dos pontos do problema. Isto nos cria duas dificuldades: o tempo, quase sempre excessivo, gasto no cálculo das distâncias e a grande quantidade de memória necessária para armazenar o vetor. O número de posições deste vetor é, aproximadamente $\frac{n^2}{2}$, no caso do PCV euclidiano, e normalmente cada posição ocupa 4 bytes, se quisermos armazenar as distâncias como números reais. Para se ter uma idéia, para uma instância

de 100 nós, o vetor ocupará em torno de 20 Kb de memória, sendo calculadas 5000 distâncias; no caso de 1000 nós, estes dados crescem para respectivamente 2 Mb e 500.000 distâncias. Como se pode notar, o crescimento quadrático destas duas variáveis pode facilmente inviabilizar a aplicação pura e simples destas heurísticas às instâncias de grande porte. Uma alternativa para reduzir o tempo dispendido no cálculo das distâncias seria estabelecer uma função mais simples que apenas estime um valor aproximado para as distâncias, evitando-se assim o cálculo de raízes quadradas ou cúbicas (caso de grafos planares e tri-dimensionais respectivamente) que são responsáveis por um consumo de tempo computacional extremamente elevado. O caminho mais correto entretanto, parece ser o de procurar evitar a necessidade de calcular grandes vetores de distâncias, de forma a não gastar um tempo excessivo antes mesmo de começar o trabalho algoritmico propriamente dito (no caso a execução da heurística). As heurísticas descritas na próxima seção seguem exatamente esta orientação.

## *II.3 - Métodos Heurísticos*

*Reinelt* [38] apresenta três enfoques diferentes para a construção de heurísticas para o PCV de grande porte e todos eles apresentam uma característica em comum: tentam reduzir o problema de modo que o(s) problema(s) resultante(s) possam ser tratados por heurísticas convencionais. Os métodos são descritos a seguir:

### *A) Redução de nós*

A idéia principal é conseguir uma significativa redução do número de nós de tal forma que os nós do problema reduzido, denominados *nós representativos,* representem satisfatoriamente a geometria do problema original. Uma vez obtida a redução, é calculado um tour pelos nós representativos, os nós originais são nele inseridos e, finalmente, os nós representativos são removidos, obtendo-se um tour para o problema original. O algoritmo sugerido por Reinelt é o seguinte:

- Obter um retângulo contendo os pontos do problema original.

- Dividir recursivamente cada retângulo em 4 retângulos de igual tamanho até que suas dimensões lineares sejam menores que $s$ % do retângulo original (ou contenha um só ponto) e que não contenha mais que $k$ pontos.

- Representar cada retângulo pelo centro de gravidade dos pontos contidos nele (o que define os nós representativos).

- Calcular um tour pelos nós representativos.

- Inserir os nós originais no tour verificando no máximo $l$ possíveis pontos de inserção.

- Remover os nós representativos não originais do tour.

A complexidade deste algoritmo é, segundo Reinelt, $O(n(\max[\log s, \log \frac{n}{k}] + l))$ acrescido do tempo necessário para calcular o tour global pelos nós representativos.

Reinelt, em seus experimentos práticos, concluiu que para obter bons resultados é muito importante obter bons tours globais pelos nós representativos. Seus melhores resultados foram colhidos empregando a heurística das economias de *Clarke e Wright* [44] para calcular o tour global e com $s = 15$ , $k = 20$ e $l = 200$ .

*B) Transformação em subgrafo esparso*

Procura minimizar uma das maiores desvantagens de quase todas as heurísticas convencionais que é a de dispender uma vasta quantidade de tempo tratando arestas "inúteis", ou seja, arestas de grande comprimento que dificilmente poderiam estar contidas num tour de qualidade razoável. O procedimento adotado é considerar um conjunto de arestas candidatas de boa qualidade, as quais são consideradas prioritárias pela heurística que vai determinar o tour. Reinelt considerou duas formas possíveis para definir este subconjunto de arestas candidatas:

- Vizinhos mais próximos - são incluídas no conjunto de arestas candidatas todas aquelas que levam qualquer nó do problema a um de seus $k$ vizinhos mais próximos. A instância de 2392 cidades estudada por *Padberg e Rinaldi* [24], por exemplo, tem a sua solução ótima contida num subconjunto de arestas de 8-vizinhos mais

próximos. Um procedimento simples de enumeração pode calcular este subconjunto de k-vizinhos em tempo $O(n^2)$ o que para instâncias grandes é um tempo muito elevado. Veremos no próximo item que é possível obter procedimentos com um comportamento esperado linear.

- Grafo de Delaunay - é uma das melhores maneiras de trabalhar com a estrutura geométrica de problemas euclidianos. Pode ser obtido através do diagrama de Voronoi que é construído da seguinte forma: para cada ponto $i$ do problema, define-se a região $V_i$ com a propriedade de que todo ponto nela contido está mais próximo do ponto $i$ do que de qualquer outro ponto do problema. O grafo (ou triangulação, como é conhecido) de Delaunay é obtido conectando-se os pontos do problema cujas regiões de Voronoi têm interseção não nula. O grafo de Delaunay pode ser calculado no pior caso em tempo $O(n \log n)$ . Para propósitos práticos, é possível obtê-lo em tempo linear com o algoritmo de *Ohya et alii* [57] cujo comportamento esperado é $O(n)$ , embora no pior caso seja $O(n^2)$ . Uma das propriedades mais interessantes do grafo de Delaunay é a de que ele contém, para todo nó, uma aresta para seu vizinho mais próximo. É fácil então concluir que é possível calcular o subgrafo de k-vizinhos mais próximos a partir do grafo de Delaunay de forma bastante simples: basta

considerar para cada ponto, os demais pontos que estejam no máximo *k* arestas afastados dele.

Reinelt obteve bons resultados, tanto em termos de tempo quanto de qualidade, com subgrafos com 10-vizinhos mais próximos. Já o grafo de Delaunay mostrou ter poucas arestas para possibilitar a busca de boas soluções com sucesso. Foi então necessário adicionar a ele as arestas transitivas de ordem 2, e, com isso, foi possível obter resultados satisfatórios. A estes dois subconjuntos de arestas candidatas foi aplicada a heurística do vizinho mais próximo (com pequenas modificações) seguida por uma otimização por 2-ótimo ou Lin-Kernighan (limitado a um máximo de 10 trocas por passo). Os resultados obtidos tanto em termos de tempo computacional quanto em qualidade foram semelhantes para os dois subconjuntos. Embora o Lin-Kernighan sempre forneça melhores soluções, o 2-ótimo revelou ter uma excelente relação tempo computacional x qualidade.

*C) Particionamento*

É uma outra forma possível de reduzir a complexidade dos problemas de grande porte. A idéia é particionar o problema em diversos subproblemas menores para poder aplicar heurísticas convencionais a cada um deles. A dificuldade é estabelecer um modo de particionar o problema de forma a que a conjunção das soluções dos subproblemas possam fornecer uma boa solução para o problema inicial. Uma vez obtido o particionamento, Reinelt sugere a seguinte heurística:

- Representar cada subconjunto pelos pontos situados em sua envoltória convexa.

- Calcular um tour global conectando os subconjuntos. Cada subconjunto é representado por um nó e a distância entre dois deles é definida pelo par de pontos mais próximos que pertencem aos dois subconjuntos. Com isto, ficam definidos pontos de entrada e saída para todos os subproblemas.

- Calcular os caminhos hamiltonianos ligando os pontos de entrada e saída de cada subproblema.

- Fazer a combinação do tour global com os caminhos hamiltonianos.

O tempo computacional requerido vai depender da forma como for feito o particionamento e das heurísticas que forem utilizadas no segundo e terceiro passos do procedimento descrito acima. Reinelt usou duas formas distintas de particionar o problema: as componentes conexas do subgrafo de 2 ou 3 vizinhos mais próximos (dependendo do número de componentes conexas induzidas) e a partição de Delaunay, obtida do grafo de Delaunay da seguinte forma:

- Classificar as arestas do grafo de Delaunay em ordem crescente de comprimento

- Examinar as arestas nesta ordem e acrescentá-las à partição se o número de componentes conexas não cair abaixo de um certo valor *l* e nenhuma componente conexa ultrapassar um determinado número *s* de arestas.

Os experimentos em que tanto o tour global quanto os caminhos hamiltonianos foram calculados com heurísticas simples (vizinho mais próximo e 2-ótimo) revelaram bons resultados para a partição de Delaunay enquanto que a partição de vizinhos mais próximos apresentou um comportamento intermitente, com resultados ora bons, ora péssimos.

As três heurísticas propostas por Reinelt obtiveram resultados razoáveis, mas ainda dispendem uma grande quantidade de tempo em algumas fases do algoritmo de transformação em subgrafo esparso (cálculo dos subgrafos de Delaunay ou de vizinhos mais próximos e otimização por Lin-Kernighan) e de particionamento (cálculo de partições e envoltórias convexas). A heurística de redução de nós também é passível de simplificações. Logo, é bem provável que seja possível encontrar heurísticas mais rápidas que obtenham soluções de qualidade não muito distante das obtidas pelas heurísticas de Reinelt. Na próxima seção discutiremos exatamente isto, ou seja, alternativas para tornar ainda mais rápido o funcionamento de heurísticas aplicáveis ao PCV de grande porte.

## *II.4 - Heurísticas Propostas*

Se analisarmos as três heurísticas descritas na seção anterior, poderemos concluir que a mais simples e rápida é a de redução de nós. Construiremos então duas heurísticas baseadas neste esquema. A primeira delas, será uma simplificação da heurística proposta por Reinelt. Nesta heurística, deixaremos de levar em conta o critério de subdivisão de retângulos baseado na relação dimensões correntes x dimensões originais. Isto irá reduzir o número máximo de nós de um retângulo e aumentar o número de retângulos gerados. Desta forma, cresce a complexidade do tour global e decresce a do procedimento posterior de inserção. Para compensar este fato, será feito um maior investimento no tour global, e, com esta estratégia, será tentada a redução do consumo de tempo computacional sem comprometer a qualidade das soluções. A segunda heurística, ainda baseada na redução de nós, terá um esquema completamente diferente de agrupamento de nós. Buscaremos, com uma heurística simples, um certo número de pontos, razoavelmente dispersos (raízes) e agruparemos os demais pontos com a raiz mais próxima deles. Este critério de agrupamento estatístico foi proposto por *Ball e Hall* [58] . Será muito interessante verificar que tipo de influência este agrupamento em torno de pontos dispersos pode ter no resultado final. A razão de utilizarmos um método heurístico simples para estabelecer o critério de dispersão é simples: maximizar a dispersão é um problema tão difícil quanto o PCV [*Erkut*, 39] e evidentemente não haveria sentido em fazer um grande esforço apenas para resolver com maior precisão um problema intermediário de menor importância. No próximo capítulo, apresentaremos então a descrição, análise e experimentos práticos com estas heurísticas.

# CAPÍTULO III

## As Heurísticas

## *III.1 - Descrição*

### *A) Redução de nós convencional*

A idéia básica da heurística de Reinelt, ou seja, a divisão recursiva de retângulos até que reste um número limitado de pontos em cada retângulo, foi mantida. A diferença fundamental entre as duas heurísticas reside no critério usado para estabelecer o limite em que a subdivisão de retângulos cessará. Na nova heurística, este limite será uma função polinomial do número de pontos do problema original. A escolha desta função, como veremos a seguir, terá grande influência na qualidade dos resultados obtidos. O algoritmo é então descrito a seguir:

- Obter um retângulo contendo os pontos do problema original

- Dividir recursivamente cada retângulo em 4 retângulos de iguais dimensões até que cada retângulo não contenha mais que $m = f(n)$ pontos, onde $n$ é o número de pontos do problema e $f$ deve ser uma função tal que $\forall n, f(n) < n$ para evitar o caso trivial (nenhuma divisão).

- Representar cada um dos $k$ retângulos pelo centro de gravidade dos pontos contidos nele.

- Calcular um tour pelos nós representativos.

- Inserir os nós originais no tour pelo método da inserção mais barata, utilizando como possíveis pontos de inserção apenas os pontos que pertencem ao mesmo retângulo.

- Remover os nós representativos.

Para o cálculo do tour pelos nós representativos, optamos por uma heurística bastante simples: tomando-se como ponto de partida cada um dos $k$ nós representativos, executamos a heurística do vizinho mais próximo seguida de uma otimização local por 2-ótimo (Uma boa descrição tanto destas heurísticas como de várias outras pode ser encontrada em *Golden et alii* [09] ). Ao final do processo, selecionamos o menor dos $k$ tours. É claro que para grandes valores de $k$ esta heurística pode se tornar inviável. Entretanto, veremos mais adiante que há meios de contornar este problema de forma satisfatória.

As figuras 1 a 5 mostram o comportamento da heurística aplicada a uma instância de 36 nós (foi intencionalmente escolhida uma instância de pequeno porte para facilitar a visualização do comportamento do algoritmo) em que foi utilizada a função $\sqrt{n}$ como limitadora do número máximo de pontos por retângulo, o que leva a um máximo de 6 nós por retângulo e por conseguinte são formados 17 retângulos. As figuras mostram o problema original, a redução do problema, o tour pelos nós representativos, o tour obtido após a inserção dos pontos originais e a solução viável obtida após a retirada dos nós representativos.

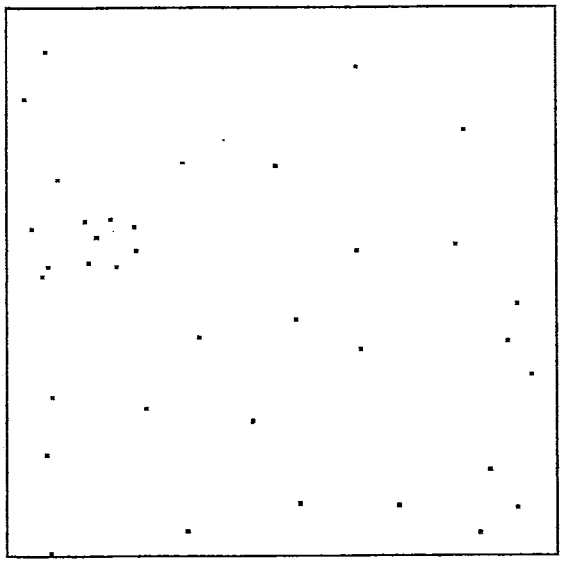

fig. 1 - problema original

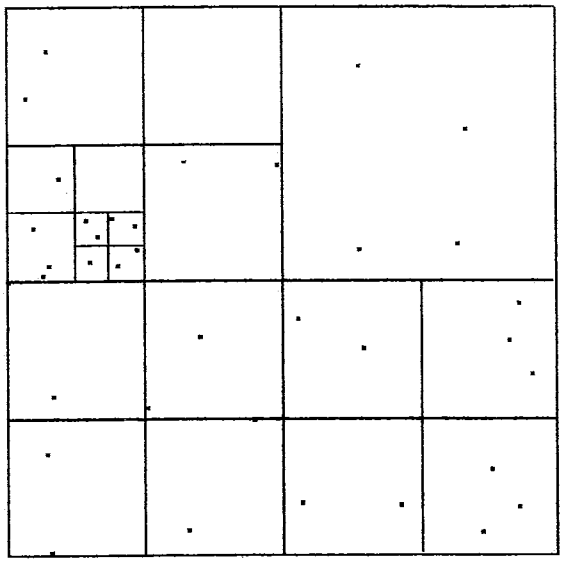

fig. 2 - ao término da divisão

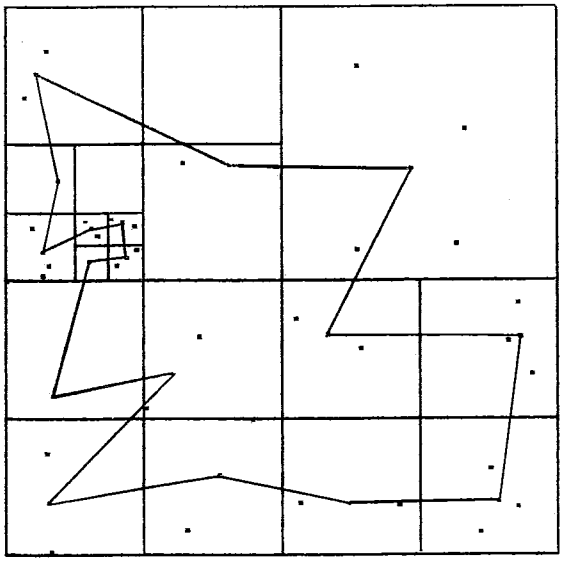

fig. 3 - tour pelos nós representativos

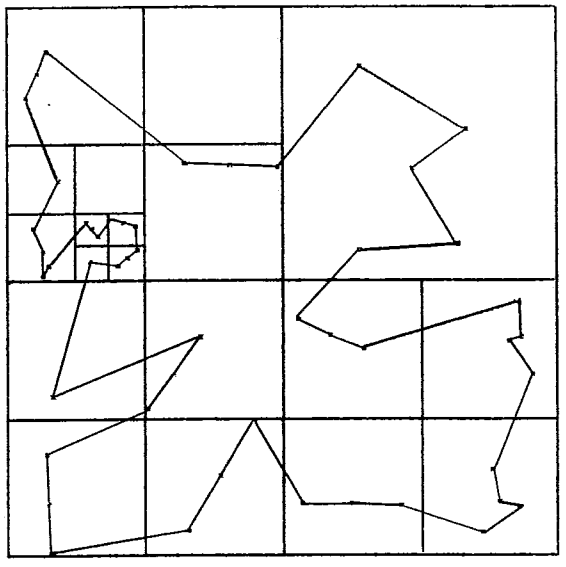

fig. 4 - inserção dos nós originais

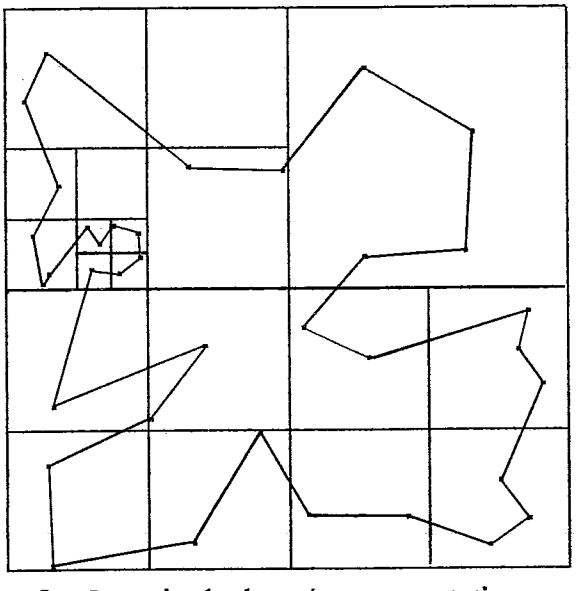

fig. 5 - retirada dos nós representativos

*B) Redução de nós por dispersão*

É evidente que a técnica utilizada para agrupar os pontos do problema original pode ter grande influência na qualidade da solução final. Uma técnica não convencional de agrupamento consiste em buscar uma certa quantidade de pontos do problema original relativamente dispersos, isto é, de forma a procurar maximizar a distância mínima entre eles, e depois utilizá-los como raízes de grupamentos ("clusters"), associando-se os demais pontos do problema ao grupamento cuja raiz for a mais próxima dele. A descrição do algoritmo é a seguinte:

- Escolher arbitrariamente o número *k* de grupamentos em que se quer dividir o problema.

- Considerando-se os *n* pontos do problema original na ordem de entrada, determinar interativamente, por pesquisa binária por exemplo, a maior distância possível para que sejam selecionadas exatamente *k* raízes.

- Uma vez obtidas as raízes, determinar os *k* grupamentos, associando cada um dos demais pontos do problema ao grupamento de raiz mais próxima.

- Representar cada um dos *k* grupamentos pelo centro de gravidade dos pontos contidos nele.

- Calcular um tour pelos nós representativos.

- Inserir os nós originais no tour pelo método da inserção mais barata, utilizando como possíveis pontos de inserção apenas os pontos que pertencem ao mesmo grupamento.

- Remover os nós representativos.

O cálculo do tour pelos nós representativos será feito de forma análoga ao da heurística de redução convencional proposta, ou seja, para cada nó representativo tomado como ponto de partida, será calculado um tour pela heurística de vizinho mais próximo que posteriormente será otimizado por 2-ótimo. Será escolhido então o menor dentre os $k$ tours calculados.

As figuras 6 a 10 ilustram o comportamento da redução por dispersão para a mesma instância de 36 nós mostrada anteriormente. Neste exemplo, arbitramos $k = 7$ grupamentos. As figuras 7, 8, 9 e 10 mostram respectivamente as raízes e seus grupamentos; o tour pelos nós representativos; o tour obtido depois da inserção dos nós originais e a solução viável obtida após a remoção dos nós representativos.

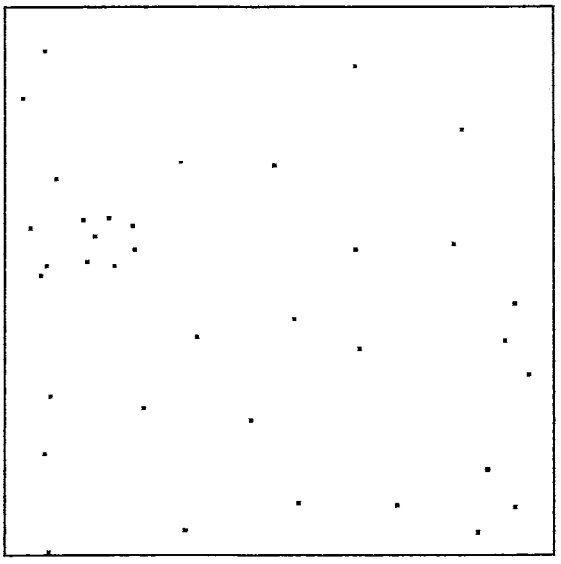

fig. 6 - problema original

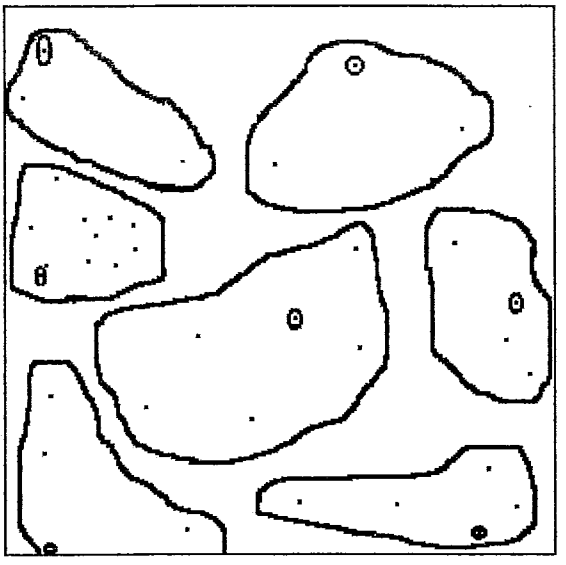

fig. 7 - raízes definem os grupamentos

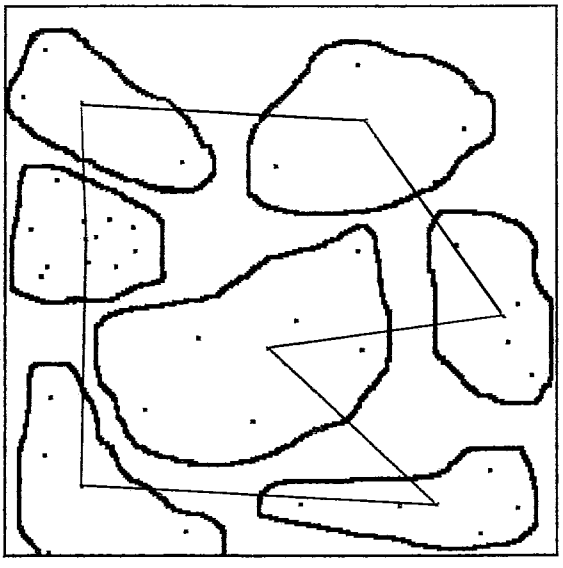

fig. 8 - tour pelos nós representativos

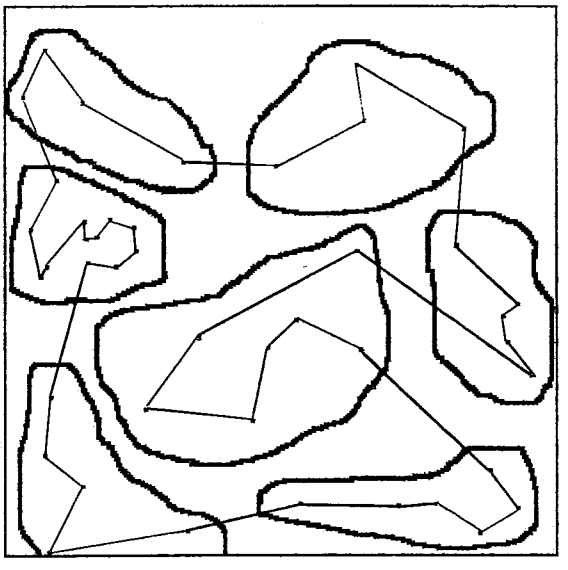

fig. 9 - inserção dos nós originais

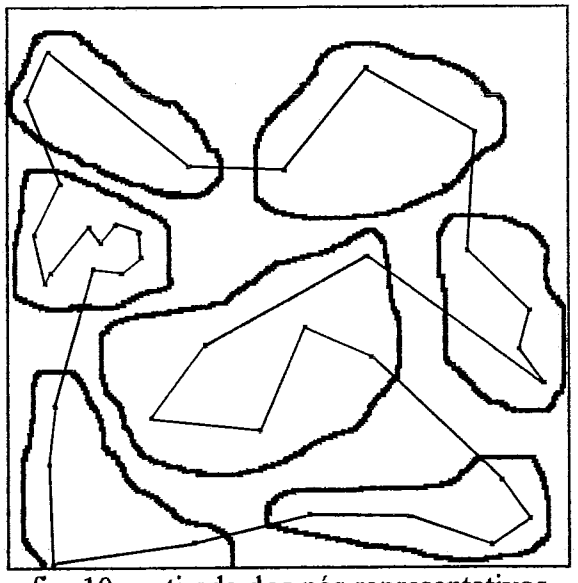

fig. 10 - retirada dos nós representativos

## *III.2 - Considerações sobre os testes*

A diferença básica entre as duas heurísticas se concentra apenas na redução inicial do problema, uma vez que nos demais passos ambas atuam de maneira semelhante. Desta forma uma comparação razoável entre elas pode ser feita se, para cada uma das instâncias utilizadas, estabelecermos que o número de retângulos gerados pela aplicação da heurística de redução convencional seja utilizado como número de grupamentos que deverão ser gerados pela heurística de redução por dispersão. A justificativa deste fato é simples: a complexidade destas heurísticas é normalmente equivalente a complexidade da determinação do tour pelos nós representativos na qual a execução da heurística 2-ótimo é o procedimento de custo computacional mais elevado. Cada passo da execução desta heurística tem complexidade $O(k^2)$ e, como o número de passos em cada corrida não pode ser determinado, podemos dizer que a complexidade do cálculo de um tour pelos nós representativos é $O(k^2 f(k))$ . Como calculamos um tour tendo cada nó representativo como ponto de partida, a complexidade final é $O(k^3 f(k))$ . A partir dos resultados práticos foi possível estimar o comportamento da heurística como $O(k^{3.8})$ , o que nos permite acreditar que o termo $f(k)$ é aproximadamente equivalente a $k^{0.8}$ . A relação entre $n$ e $k$ é então de fundamental importância para a eficiente utilização destas heurísticas. Por exemplo, se $n = k$ (caso trivial onde não há redução do problema), a complexidade da heurística será $O(n^{3.8})$ , o que para grandes instâncias será inviável. Por outro lado, se $k = \sqrt{n}$ , a complexidade será menor que $O(n^2)$ , o que pode na maioria casos, ser aceitável.

Outros fatores importantes são:

- Escolha da função $f(n)$ que limitará o número máximo de pontos num retângulo. Tem influência direta no número de nós representativos gerado e, desta forma, na complexidade da heurística. Para verificar como a heurística se comportaria num espectro mais amplo possível, escolhemos 8 funções: $3\sqrt{n}$, $2\sqrt{n}$, $3\frac{\sqrt{n}}{2}$, $\sqrt{n}$, $\frac{\sqrt{n}}{2}$, $\frac{\sqrt{n}}{3}$, $\frac{\sqrt{n}}{4}$ e $\frac{\sqrt{n}}{8}$. Como as duas últimas funções levaram ao gasto de tempos computacionais expressivos devido ao grande número $k$ de nós representativos (pois para cada um deles tomado como ponto de partida calcula-se um tour seguido de uma otimização por 2-ótimo), fizemos também uma tentativa para reduzir o tempo computacional destas funções tomando o tour global como o melhor dentre os primeiros $\sqrt{k}$ nós representativos tomados como ponto de partida, desprezando-se então o cálculo de $k-\sqrt{k}$ tours. Com este procedimento o gasto de cpu sofre uma redução bastante significativa e a qualidade das soluções não sofre grandes alterações como veremos na próxima seção.

- Instâncias utilizadas: Para podermos mais tarde fazer uma boa avaliação da qualidade das soluções fornecidas pelas heurísticas, o ideal seria utilizar nos testes instâncias cujas soluções ótimas já fossem conhecidas. Porém, tais instâncias são raríssimas na

literatura. Em 1985, *Arthur e Frendewey* [61] . propuseram um algoritmo capaz de gerar matrizes de distâncias de instâncias cuja solução ótima pode ser facilmente calculada. Essas instâncias podem ser assimétricas, simétricas e até euclidianas; entretanto as heurísticas aqui propostas não podem utilizá-las, pois como são baseadas fundamentalmente na geometria do problema, necessitam de um vetor de coordenadas como entrada. O cálculo de um vetor de coordenadas que corresponda à matriz de distâncias gerada pelo algoritmo parece ser um problema bastante complexo (provavelmente não polinomial) e está fora do escopo deste trabalho. Algoritmos que gerem vetores de coordenadas de instâncias com solução ótima facilmente calculável são desconhecidos na literatura. Desta forma, optamos por usar nos testes em sua maioria instâncias geradas aleatoriamente no quadrado de lado 1 que podem ser objeto de avaliação estatística. Foram incluidas instâncias desde 100 até 10000 nós. Com exceção da instância de 318 cidades de Lin e Kernighan [08] , na qual não foram considerados fixos os pontos de início e fim que constam do problema original, todas as outras instâncias são aleatórias. Não foram incluidas instâncias reais de maior porte com solução ótima conhecida como as de *Padberg e Rinaldi* [13, 24] devido à dificuldade de obtê-las.

- Os programas foram construídos em linguagem C e a máquina utilizada para os testes foi uma estação RISC 6000 mod. 530H com 20.5 MFlops e 32 Mb de memória em ambiente AIX.

## *III.3 - Resultados Obtidos*

As tabelas a seguir mostram os resultados alcançados pelas heurísticas para cada instância considerada. As instâncias C100A, C100B, C100C e C100D têm 100 nós; C200A tem duzentos e etc. Assim as instâncias vão sendo apresentadas em ordem crescente de número de nós até a instância C10000A. O conteúdo das colunas é o seguinte:

1 - função utilizada para limitação de pontos no retângulo

2 - número de retângulos gerado pela aplicação da heurística de redução convencional = número de grupamentos empregado na redução por dispersão

3 - ciclo obtido pela aplicação da redução convencional

4 - tempo de cpu gasto pela heurística de redução convencional

5 - ciclo obtido pela aplicação da redução por dispersão

6 - tempo de cpu gasto pela heurística de redução por dispersão

As duas últimas linhas correspondem às funções $\frac{\sqrt{n}}{4}$ e $\frac{\sqrt{n}}{8}$ nas quais foram calculados apenas $\sqrt{k}$ tours pelos nós representativos.

As células vazias indicam que a execução não terminou nem mesmo em 80 horas de cpu.

| C100A | grupamentos | TOUR 1 | TEMPO(s) | TOUR 2 | TEMPO(s) |
|---|---|---|---|---|---|
| $3\sqrt{n}$ | 4 | 8,78273 | 0,0 | 8,74036 | 0,0 |
| $2\sqrt{n}$ | 16 | 9,17108 | 0,0 | 9,03927 | 0,1 |
| $3\frac{\sqrt{n}}{2}$ | 16 | 9,17108 | 0,0 | 9,03927 | 0,1 |
| $\sqrt{n}$ | 19 | 9,13663 | 0,0 | 8,63659 | 0,1 |
| $\frac{\sqrt{n}}{2}$ | 37 | 8,71496 | 0,6 | 8,27776 | 0,5 |
| $\frac{\sqrt{n}}{3}$ | 55 | 8,10629 | 1,7 | 8,12156 | 2,0 |
| $\frac{\sqrt{n}}{4}$ | 70 | 8,06129 | 4,7 | 7,93684 | 6,4 |
| $\frac{\sqrt{n}}{8}$ | 100 | 7,92667 | 24,5 | 7,92421 | 23,8 |
| $\frac{\sqrt{n}}{4}$ *** | 70 | 8,18096 | 0,3 | 7,93684 | 0,9 |
| $\frac{\sqrt{n}}{8}$ *** | 100 | 8,00348 | 2,2 | 8,00553 | 2,5 |

| C100B | grupamentos | TOUR 1 | TEMPO(s) | TOUR 2 | TEMPO(s) |
|---|---|---|---|---|---|
| $3\sqrt{n}$ | 4 | 7,77456 | 0,0 | 8,35250 | 0,0 |
| $2\sqrt{n}$ | 13 | 7,78870 | 0,0 | 8,39633 | 0,0 |
| $3\frac{\sqrt{n}}{2}$ | 16 | 7,84604 | 0,0 | 7,99505 | 0,0 |
| $\sqrt{n}$ | 19 | 7,84021 | 0,0 | 8,17975 | 0,1 |
| $\frac{\sqrt{n}}{2}$ | 40 | 8,15141 | 0,6 | 7,70658 | 0,6 |
| $\frac{\sqrt{n}}{3}$ | 59 | 7,77499 | 1,7 | 7,52732 | 2,3 |
| $\frac{\sqrt{n}}{4}$ | 68 | 7,66188 | 6,0 | 7,50153 | 4,4 |
| $\frac{\sqrt{n}}{8}$ | 100 | 7,55365 | 27,3 | 7,55703 | 21,4 |
| $\frac{\sqrt{n}}{4}$ *** | 68 | 7,66188 | 0,8 | 7,67517 | 0,9 |
| $\frac{\sqrt{n}}{8}$ *** | 100 | 7,57258 | 2,1 | 7,61361 | 2,2 |

| C100C | grupamentos | TOUR 1 | TEMPO(s) | TOUR 2 | TEMPO(s) |
|---|---|---|---|---|---|
| $3\sqrt{n}$ | 7 | 8,53540 | 0,0 | 8,87415 | 0,0 |
| $2\sqrt{n}$ | 13 | 8,39779 | 0,0 | 8,60896 | 0,0 |
| $3\frac{\sqrt{n}}{2}$ | 16 | 8,42245 | 0,0 | 8,37199 | 0,1 |
| $\sqrt{n}$ | 22 | 8,28876 | 0,0 | 8,05424 | 0,1 |
| $\frac{\sqrt{n}}{2}$ | 37 | 8,34147 | 0,4 | 8,18832 | 0,6 |
| $\frac{\sqrt{n}}{3}$ | 57 | 7,95179 | 2,6 | 7,84803 | 2,3 |
| $\frac{\sqrt{n}}{4}$ | 72 | 7,62863 | 5,4 | 7,63327 | 5,7 |
| $\frac{\sqrt{n}}{8}$ | 100 | 7,58401 | 27,3 | 7,71417 | 16,9 |
| $\frac{\sqrt{n}}{4}$ *** | 72 | 8,01792 | 0,4 | 7,77076 | 0,8 |
| $\frac{\sqrt{n}}{8}$ *** | 100 | 7,68754 | 1,8 | 7,75352 | 1,8 |

| C100D | grupamentos | TOUR 1 | TEMPO(s) | TOUR 2 | TEMPO(s) |
|---|---|---|---|---|---|
| $3\sqrt{n}$ | 4 | 8,38021 | 0,0 | 9,35579 | 0,0 |
| $2\sqrt{n}$ | 13 | 8,75914 | 0,0 | 8,90470 | 0,0 |
| $3\frac{\sqrt{n}}{2}$ | 16 | 8,87783 | 0,0 | 9,01245 | 0,0 |
| $\sqrt{n}$ | 16 | 8,87783 | 0,0 | 9,01245 | 0,0 |
| $\frac{\sqrt{n}}{2}$ | 42 | 8,44358 | 0,8 | 7,95996 | 0,8 |
| $\frac{\sqrt{n}}{3}$ | 61 | 8,30067 | 3,5 | 7,95002 | 4,3 |
| $\frac{\sqrt{n}}{4}$ | 74 | 7,97842 | 8,2 | 7,94853 | 6,9 |
| $\frac{\sqrt{n}}{8}$ | 100 | 7,87544 | 20,7 | 7,87544 | 18,1 |
| $\frac{\sqrt{n}}{4}$ *** | 74 | 7,98327 | 0,8 | 8,18497 | 0,8 |
| $\frac{\sqrt{n}}{8}$ *** | 100 | 7,92887 | 1,8 | 7,92076 | 1,8 |

| C200A | grupamentos | TOUR 1 | TEMPO(s) | TOUR 2 | TEMPO(s) |
|---|---|---|---|---|---|
| $3\sqrt{n}$ | 16 | 12,82380 | 0,0 | 12,41172 | 0,1 |
| $2\sqrt{n}$ | 16 | 12,82380 | 0,0 | 12,41172 | 0,1 |
| $3\sqrt{\frac{n}{2}}$ | 16 | 12,82380 | 0,0 | 12,41172 | 0,1 |
| $\sqrt{n}$ | 28 | 12,61972 | 0,2 | 12,75914 | 0,2 |
| $\sqrt{\frac{n}{2}}$ | 63 | 12,42583 | 4,0 | 11,95543 | 2,8 |
| $\sqrt{\frac{n}{3}}$ | 86 | 12,32317 | 11,9 | 11,79064 | 7,8 |
| $\sqrt{\frac{n}{4}}$ | 116 | 12,18265 | 39,6 | 11,56601 | 44,9 |
| $\sqrt{\frac{n}{8}}$ | 200 | 11,31104 | 333,1 | 11,29702 | 349,1 |
| $\sqrt{\frac{n}{4}}$ *** | 116 | 12,16004 | 2,5 | 11,68151 | 4,9 |
| $\sqrt{\frac{n}{8}}$ *** | 200 | 11,36387 | 22,0 | 11,39659 | 25,6 |

| C300A | grupamentos | TOUR 1 | TEMPO(s) | TOUR 2 | TEMPO(s) |
|---|---|---|---|---|---|
| $3\sqrt{n}$ | 16 | 16,07904 | 0,0 | 16,04279 | 0,1 |
| $2\sqrt{n}$ | 16 | 16,07904 | 0,0 | 16,04279 | 0,1 |
| $3\sqrt{\frac{n}{2}}$ | 25 | 15,66243 | 0,1 | 16,06788 | 0,2 |
| $\sqrt{n}$ | 40 | 15,88172 | 0,8 | 15,50460 | 0,9 |
| $\sqrt{\frac{n}{2}}$ | 71 | 15,20205 | 6,2 | 14,67092 | 5,2 |
| $\sqrt{\frac{n}{3}}$ | 120 | 14,66685 | 36,5 | 13,88774 | 35,6 |
| $\sqrt{\frac{n}{4}}$ | 143 | 14,43256 | 83,7 | 13,83588 | 92,6 |
| $\sqrt{\frac{n}{8}}$ | 219 | 13,90309 | 572,1 | 13,53258 | 493,6 |
| $\sqrt{\frac{n}{4}}$ *** | 143 | 14,69049 | 5,3 | 14,22240 | 7,8 |
| $\sqrt{\frac{n}{8}}$ *** | 219 | 14,04929 | 29,0 | 13,53258 | 32,1 |

| C318A | grupamentos | TOUR 1 | TEMPO(s) | TOUR 2 | TEMPO(s) |
|---|---|---|---|---|---|
| $3\sqrt{n}$ | 16 | 52.478,96 | 0,1 | 53.423,11 | 0,2 |
| $2\sqrt{n}$ | 16 | 52.478,96 | 0,1 | 53.423,11 | 0,2 |
| $3\frac{\sqrt{n}}{2}$ | 22 | 52.146,16 | 0,1 | 49.081,15 | 0,2 |
| $\sqrt{n}$ | 52 | 50.233,77 | 2,5 | 46.026,59 | 2,0 |
| $\frac{\sqrt{n}}{2}$ | 90 | 47.693,90 | 15,9 | 44.422,14 | 14,6 |
| $\frac{\sqrt{n}}{3}$ | 136 | 45.584,09 | 65,7 | 43.951,52 | 78,2 |
| $\frac{\sqrt{n}}{4}$ | 154 | 45.455,52 | 131,0 | 43.824,60 | 128,6 |
| $\frac{\sqrt{n}}{8}$ | 215 | 43.904,00 | 396,4 | 43.791,00 | 419,6 |
| $\frac{\sqrt{n}}{4}$ *** | 154 | 45.771,87 | 10,0 | 44.168,97 | 12,9 |
| $\frac{\sqrt{n}}{8}$ *** | 215 | 44.245,13 | 28,2 | 44.303,16 | 25,4 |

| C400A | grupamentos | TOUR 1 | TEMPO(s) | TOUR 2 | TEMPO(s) |
|---|---|---|---|---|---|
| $3\sqrt{n}$ | 16 | 17,59017 | 0,1 | 17,83217 | 0,2 |
| $2\sqrt{n}$ | 16 | 17,59017 | 0,1 | 17,83217 | 0,2 |
| $3\frac{\sqrt{n}}{2}$ | 22 | 17,65346 | 0,1 | 17,56488 | 0,2 |
| $\sqrt{n}$ | 55 | 17,32030 | 1,7 | 17,09322 | 1,9 |
| $\frac{\sqrt{n}}{2}$ | 72 | 17,71031 | 6,6 | 16,81329 | 5,5 |
| $\frac{\sqrt{n}}{3}$ | 134 | 16,86059 | 54,1 | 15,56264 | 64,8 |
| $\frac{\sqrt{n}}{4}$ | 163 | 16,36758 | 139,9 | 15,31517 | 127,0 |
| $\frac{\sqrt{n}}{8}$ | 297 | 15,48508 | 1.486,5 | 15,31908 | 1.207,4 |
| $\frac{\sqrt{n}}{4}$ *** | 163 | 16,90705 | 10,0 | 15,54115 | 9,9 |
| $\frac{\sqrt{n}}{8}$ *** | 297 | 15,67310 | 73,6 | 15,35380 | 71,7 |

| C500A | grupamentos | TOUR 1 | TEMPO(s) | TOUR 2 | TEMPO(s) |
|---|---|---|---|---|---|
| $3\sqrt{n}$ | 16 | 19,77919 | 0,1 | 19,31718 | 0,3 |
| $2\sqrt{n}$ | 16 | 19,77919 | 0,1 | 19,31718 | 0,3 |
| $3\sqrt{\frac{n}{2}}$ | 34 | 19,44788 | 0,4 | 19,07858 | 0,7 |
| $\sqrt{n}$ | 58 | 19,13688 | 2,6 | 19,55712 | 2,8 |
| $\frac{\sqrt{n}}{2}$ | 82 | 19,68990 | 9,2 | 18,97020 | 8,3 |
| $\frac{\sqrt{n}}{3}$ | 151 | 19,03395 | 76,4 | 17,92572 | 102,1 |
| $\frac{\sqrt{n}}{4}$ | 206 | 18,17307 | 360,8 | 17,49955 | 300,5 |
| $\frac{\sqrt{n}}{8}$ | 368 | 17,42568 | 3.746,9 | 17,14126 | 3.673,1 |
| $\frac{\sqrt{n}}{4}$ *** | 206 | 18,40519 | 19,7 | 17,55729 | 22,0 |
| $\frac{\sqrt{n}}{8}$ *** | 368 | 17,48018 | 160,5 | 17,14126 | 183,6 |

| C1000A | grupamentos | TOUR 1 | TEMPO(s) | TOUR 2 | TEMPO(s) |
|---|---|---|---|---|---|
| $3\sqrt{n}$ | 16 | 26,50940 | 0,6 | 26,27310 | 0,9 |
| $2\sqrt{n}$ | 37 | 26,73386 | 0,7 | 26,99128 | 1,1 |
| $3\sqrt{\frac{n}{2}}$ | 64 | 27,42570 | 2,7 | 27,13591 | 4,5 |
| $\sqrt{n}$ | 64 | 27,42570 | 2,5 | 27,13591 | 4,5 |
| $\frac{\sqrt{n}}{2}$ | 163 | 26,85942 | 123,8 | 26,67767 | 141,8 |
| $\frac{\sqrt{n}}{3}$ | 235 | 26,26118 | 599,2 | 25,05256 | 474,6 |
| $\frac{\sqrt{n}}{4}$ | 267 | 26,30006 | 853,3 | 24,61111 | 753,5 |
| $\frac{\sqrt{n}}{8}$ | 555 | 24,48665 | 15.298,8 | 23,67862 | 16.313,6 |
| $\frac{\sqrt{n}}{4}$ *** | 267 | 26,11025 | 52,9 | 24,61111 | 53,2 |
| $\frac{\sqrt{n}}{8}$ *** | 555 | 24,89310 | 647,9 | 23,58860 | 675,8 |

| C1000B | grupamentos | TOUR 1 | TEMPO(s) | TOUR 2 | TEMPO(s) |
|---|---|---|---|---|---|
| $3\sqrt{n}$ | 16 | 27,40827 | 0,6 | 26,98387 | 0,9 |
| $2\sqrt{n}$ | 40 | 27,57338 | 0,9 | 27,44192 | 1,2 |
| $3\frac{\sqrt{n}}{2}$ | 64 | 27,79485 | 3,4 | 27,45596 | 4,0 |
| $\sqrt{n}$ | 64 | 27,79485 | 3,3 | 27,45596 | 4,0 |
| $\frac{\sqrt{n}}{2}$ | 166 | 27,87711 | 146,2 | 26,63271 | 155,6 |
| $\frac{\sqrt{n}}{3}$ | 228 | 27,11597 | 536,8 | 25,74011 | 419,1 |
| $\frac{\sqrt{n}}{4}$ | 282 | 26,80436 | 1.005,2 | 25,50768 | 917,4 |
| $\frac{\sqrt{n}}{8}$ | 571 | 25,40016 | 18.898,0 | 24,18551 | 15.300,8 |
| $\frac{\sqrt{n}}{4}$ AAA | 282 | 26,83716 | 47,2 | 25,64922 | 57,2 |
| $\frac{\sqrt{n}}{8}$ AAA | 571 | 25,40016 | 746,7 | 24,31943 | 629,2 |

| C1000C | grupamentos | TOUR 1 | TEMPO(s) | TOUR 2 | TEMPO(s) |
|---|---|---|---|---|---|
| $3\sqrt{n}$ | 16 | 26,88548 | 0,6 | 26,75751 | 0,9 |
| $2\sqrt{n}$ | 40 | 27,04320 | 1,0 | 26,97283 | 1,3 |
| $3\frac{\sqrt{n}}{2}$ | 64 | 26,84972 | 2,9 | 27,31974 | 4,8 |
| $\sqrt{n}$ | 64 | 26,84972 | 3,0 | 27,31974 | 4,8 |
| $\frac{\sqrt{n}}{2}$ | 162 | 26,78509 | 115,7 | 25,99295 | 129,8 |
| $\frac{\sqrt{n}}{3}$ | 234 | 26,43318 | 445,5 | 25,07839 | 503,6 |
| $\frac{\sqrt{n}}{4}$ | 280 | 26,05836 | 1.029,3 | 24,76630 | 1.036,0 |
| $\frac{\sqrt{n}}{8}$ | 557 | 24,77996 | 14.639,8 | 23,93896 | 17.171,9 |
| $\frac{\sqrt{n}}{4}$ AAA | 280 | 25,80534 | 59,6 | 25,25712 | 61,8 |
| $\frac{\sqrt{n}}{8}$ AAA | 557 | 24,79322 | 623,0 | 24,15774 | 770,0 |

| C1500A | grupamentos | TOUR 1 | TEMPO(s) | TOUR 2 | TEMPO(s) |
|---|---|---|---|---|---|
| $3\sqrt{n}$ | 16 | 32,95806 | 1,5 | 32,73548 | 1,9 |
| $2\sqrt{n}$ | 61 | 33,80906 | 3,7 | 33,85346 | 3,8 |
| $3\frac{\sqrt{n}}{2}$ | 64 | 33,85478 | 5,4 | 33,89397 | 4,7 |
| $\sqrt{n}$ | 64 | 33,85478 | 5,3 | 33,89397 | 4,7 |
| $\frac{\sqrt{n}}{2}$ | 220 | 34,20157 | 379,9 | 33,22841 | 332,6 |
| $\frac{\sqrt{n}}{3}$ | 261 | 33,96774 | 868,3 | 32,56912 | 759,8 |
| $\frac{\sqrt{n}}{4}$ | 296 | 33,60057 | 1.545,2 | 31,86799 | 1.391,5 |
| $\frac{\sqrt{n}}{8}$ | 729 | 31,78547 | 47.522,4 | 29,79461 | 42.015,1 |
| $\frac{\sqrt{n}}{4}$ ... | 296 | 33,62996 | 77,3 | 32,31001 | 84,4 |
| $\frac{\sqrt{n}}{8}$ ... | 729 | 31,91100 | 1.850,4 | 29,88311 | 1.499,5 |

| C2000A | grupamentos | TOUR 1 | TEMPO(s) | TOUR 2 | TEMPO(s) |
|---|---|---|---|---|---|
| $3\sqrt{n}$ | 22 | 38,27789 | 5,2 | 37,93409 | 2,7 |
| $2\sqrt{n}$ | 64 | 39,11020 | 12,4 | 38,96122 | 5,2 |
| $3\frac{\sqrt{n}}{2}$ | 64 | 39,11020 | 12,4 | 38,96122 | 5,2 |
| $\sqrt{n}$ | 64 | 39,11020 | 12,6 | 38,96122 | 5,2 |
| $\frac{\sqrt{n}}{2}$ | 250 | 38,97035 | 2.702,5 | 38,57435 | 636,0 |
| $\frac{\sqrt{n}}{3}$ | 268 | 39,07014 | 1.031,8 | 38,09833 | 905,4 |
| $\frac{\sqrt{n}}{4}$ | 315 | 39,12077 | 1.953,5 | 38,16658 | 1.629,2 |
| $\frac{\sqrt{n}}{8}$ | 854 | 36,55554 | 83.533,2 | 34,26983 | 82.064,5 |
| $\frac{\sqrt{n}}{4}$ ... | 315 | 39,20543 | 111,3 | 37,82379 | 97,5 |
| $\frac{\sqrt{n}}{8}$ ... | 854 | 36,71170 | 2.642,2 | 34,57363 | 2.839,0 |

| C2500A | grupamentos | TOUR 1 | TEMPO(s) | TOUR 2 | TEMPO(s) |
|---|---|---|---|---|---|
| $3\sqrt{n}$ | 43 | 42,99313 | 2,7 | 43,16755 | 2,7 |
| $2\sqrt{n}$ | 64 | 43,74313 | 3,8 | 44,22372 | 5,9 |
| $3\frac{\sqrt{n}}{2}$ | 64 | 43,74313 | 3,8 | 44,22372 | 5,9 |
| $\sqrt{n}$ | 76 | 43,92791 | 8,0 | 43,89726 | 10,3 |
| $\frac{\sqrt{n}}{2}$ | 253 | 44,14644 | 585,3 | 43,93966 | 612,0 |
| $\frac{\sqrt{n}}{3}$ | 271 | 44,32091 | 880,4 | 43,80948 | 895,3 |
| $\frac{\sqrt{n}}{4}$ | 398 | 43,87743 | 3.994,3 | 42,85285 | 3.735,7 |
| $\frac{\sqrt{n}}{8}$ | 892 | 41,67315 | 95.882,8 | 39,45732 | 94.023,2 |
| $\frac{\sqrt{n}}{4}$ AAA | 398 | 43,62137 | 243,9 | 42,86597 | 185,5 |
| $\frac{\sqrt{n}}{8}$ AAA | 892 | 42,03379 | 3.067,5 | 39,49308 | 3.168,2 |

| C3000A | grupamentos | TOUR 1 | TEMPO(s) | TOUR 2 | TEMPO(s) |
|---|---|---|---|---|---|
| $3\sqrt{n}$ | 58 | 45,93680 | 3,8 | 46,07871 | 5,7 |
| $2\sqrt{n}$ | 64 | 46,26516 | 3,4 | 45,81277 | 7,3 |
| $3\frac{\sqrt{n}}{2}$ | 64 | 46,26516 | 3,5 | 45,81277 | 7,3 |
| $\sqrt{n}$ | 94 | 46,75345 | 12,9 | 46,26338 | 15,5 |
| $\frac{\sqrt{n}}{2}$ | 256 | 46,58139 | 636,2 | 46,18898 | 751,4 |
| $\frac{\sqrt{n}}{3}$ | 295 | 46,74993 | 1.093,6 | 46,01408 | 1.404,5 |
| $\frac{\sqrt{n}}{4}$ | 492 | 46,04538 | 9.362,2 | 44,71350 | 9.093,7 |
| $\frac{\sqrt{n}}{8}$ | 1.029 | 43,94024 | 171.121,3 | 41,23352 | 151.851,31 |
| $\frac{\sqrt{n}}{4}$ AAA | 492 | 46,35554 | 543,2 | 44,85836 | 458,9 |
| $\frac{\sqrt{n}}{8}$ AAA | 1.029 | 44,22420 | 4.837,5 | 41,34990 | 4.684,5 |

| C4000A | grupamentos | TOUR 1 | TEMPO(s) | TOUR 2 | TEMPO(s) |
|---|---|---|---|---|---|
| $3\sqrt{n}$ | 64 | 53,35807 | 6,2 | 52,99336 | 9,2 |
| $2\sqrt{n}$ | 64 | 53,35807 | 6,2 | 52,99336 | 9,2 |
| $3\frac{\sqrt{n}}{2}$ | 64 | 53,35807 | 6,4 | 52,99336 | 9,2 |
| $\sqrt{n}$ | 148 | 53,87370 | 110,7 | 54,70118 | 102,7 |
| $\frac{\sqrt{n}}{2}$ | 256 | 54,63923 | 674,1 | 54,65866 | 913,6 |
| $\frac{\sqrt{n}}{3}$ | 337 | 54,80052 | 2.113,5 | 54,74612 | 2.336,5 |
| $\frac{\sqrt{n}}{4}$ | 633 | 54,18261 | 23.191,6 | 52,40150 | 22.971,1 |
| $\frac{\sqrt{n}}{8}$ | 1.141 | 52,35419 | 234.888,7 | 49,53274 | 237.392,5 |
| $\frac{\sqrt{n}}{4}$ AAA | 633 | 54,09186 | 1.070,7 | 52,30062 | 959,8 |
| $\frac{\sqrt{n}}{8}$ AAA | 1.141 | 52,62911 | 7.715,6 | 49,50778 | 6.821,6 |

| C5000A | grupamentos | TOUR 1 | TEMPO(s) | TOUR 2 | TEMPO(s) |
|---|---|---|---|---|---|
| $3\sqrt{n}$ | 64 | 59,35562 | 5,6 | 59,91882 | 12,4 |
| $2\sqrt{n}$ | 64 | 59,35562 | 5,6 | 59,91882 | 12,4 |
| $3\frac{\sqrt{n}}{2}$ | 67 | 59,08562 | 5,7 | 59,88579 | 12,5 |
| $\sqrt{n}$ | 214 | 60,90413 | 324,5 | 60,87710 | 389,8 |
| $\frac{\sqrt{n}}{2}$ | 259 | 61,06880 | 790,8 | 60,92519 | 690,3 |
| $\frac{\sqrt{n}}{3}$ | 736 | 60,94955 | 44.395,6 | 58,58672 | 41.891,4 |
| $\frac{\sqrt{n}}{4}$ | 736 | 60,94955 | 44.640,5 | 58,58672 | 41.891,4 |
| $\frac{\sqrt{n}}{8}$ | 1.205 | 59,48436 | 278.109,9 | 56,10799 | 283.052,4 |
| $\frac{\sqrt{n}}{4}$ AAA | 736 | 60,88312 | 1.663,2 | 58,71238 | 1.578,3 |
| $\frac{\sqrt{n}}{8}$ AAA | 1.205 | 59,39497 | 7.443,5 | 56,03393 | 8.175,0 |

| C8000A | grupamentos | TOUR 1 | TEMPO(s) | TOUR 2 | TEMPO(s) |
|---|---|---|---|---|---|
| $3\sqrt{n}$ | 64 | 73,65272 | 14,2 | 73,83715 | 22,0 |
| $2\sqrt{n}$ | 64 | 73,65272 | 14,1 | 73,83715 | 22,0 |
| $3\frac{\sqrt{n}}{2}$ | 112 | 74,52864 | 38,0 | 74,10943 | 40,5 |
| $\sqrt{n}$ | 250 | 76,30577 | 690,2 | 76,26034 | 703,8 |
| $\frac{\sqrt{n}}{2}$ | 265 | 76,19964 | 743,6 | 76,43890 | 799,7 |
| $\frac{\sqrt{n}}{3}$ | 703 | 76,75168 | 37.666,5 | 76,00675 | 32.846,8 |
| $\frac{\sqrt{n}}{4}$ | 961 | 76,17702 | 118.580,4 | 74,87142 | 124.062,9 |
| $\frac{\sqrt{n}}{8}$ | - | - | - | - | - |
| $\frac{\sqrt{n}}{4}$ AAA | 961 | 76,01342 | 3.564,4 | 74,86462 | 4.206,6 |
| $\frac{\sqrt{n}}{8}$ AAA | 1.378 | 75,89044 | 13.830,4 | 73,05650 | 11.363,5 |

| C10000A | grupamentos | TOUR 1 | TEMPO(s) | TOUR 2 | TEMPO(s) |
|---|---|---|---|---|---|
| $3\sqrt{n}$ | 64 | 79,83399 | 60,6 | 79,29124 | 30,4 |
| $2\sqrt{n}$ | 64 | 79,83399 | 57,8 | 79,29124 | 30,4 |
| $3\frac{\sqrt{n}}{2}$ | 175 | 81,67705 | 682,6 | 81,31780 | 195,6 |
| $\sqrt{n}$ | 256 | 82,49243 | 3.005,0 | 81,79366 | 830,4 |
| $\frac{\sqrt{n}}{2}$ | 289 | 82,61454 | 4.767,5 | 82,62332 | 1.240,6 |
| $\frac{\sqrt{n}}{3}$ | 853 | 83,09666 | 335.848,8 | 82,42190 | 80.323,40 |
| $\frac{\sqrt{n}}{4}$ | - | - | - | - | - |
| $\frac{\sqrt{n}}{4}$ | - | - | - | - | - |
| $\frac{\sqrt{n}}{4}$ AAA | 1.006 | 83,36890 | 4.987,7 | 81,69908 | 4.455,8 |
| $\frac{\sqrt{n}}{8}$ AAA | 1.690 | 82,08666 | 27.431,6 | 79,06766 | 25.873,8 |

# CAPÍTULO IV

## Análise dos Resultados

## *IV.1 - Fatores Relevantes*

Neste capítulo, vamos procurar fazer uma análise o mais abrangente possível do comportamento das duas heurísticas, tanto comparativamente, ou seja, confrontando os resultados de uma contra a outra, quanto em termos qualitativos (o que significa estimar a que distância do ciclo ótimo está a solução heurística) e de viabilidade (relação custo/benefício). Além disso, um outro aspecto importante que deve ser observado é a influência do número de grupamentos (que é definido diretamente na redução por dispersão e indiretamente pela função limitadora de pontos por grupamento na redução convencional) sobre a qualidade e custo computacional da solução obtida.

## *IV.2 - Viabilidade*

As duas heurísticas demonstraram que podem fornecer soluções com custo computacional que varia de 30 segundos (o que é relativamente baixo) para a instância de 10000 cidades (a maior de todas as que o teste foi composto) até valores inaceitáveis na prática visto que não foi possível executar esta instância com 1000 grupamentos nem mesmo em 80 horas de cpu. Como já se poderia prever, verifica-se um crescimento bastante significativo do consumo de cpu quando se experimenta um crescimento relativamente pequeno do número de grupamentos. Por exemplo, no problema C1000A, um aumento de 163 para 555 grupamentos (uma razão de 3.4) implicou em um tempo de cpu 123 e 115 vezes maior na redução convencional e na redução por dispersão respectivamente. Entretanto, podemos notar que desde que o número de grupamentos seja

convenientemente escolhido, é sempre possível obter uma solução em tempo computacional viável. O exame da qualidade destas soluções será feito na seção IV.4 .

## *IV.3 - Comparação entre as heurísticas*

O consumo de cpu das duas heurísticas não apresenta diferenças relevantes quando se compara o mesmo número de grupamentos. Na maioria das corridas a diferença não ultrapassa 30%, o que, normalmente não é significativo. Como já foi explicado na seção III.2 , a complexidade das duas heurísticas é função unicamente do número de grupamentos, e portanto, a avaliação prática apenas veio a confirmar este fato. Quanto à qualidade das soluções, podemos verificar uma nítida tendência de superioridade em favor da heurística de redução por dispersão tanto no problema real de 318 cidades quanto nos problemas aleatórios. Isto somente não ocorreu nos problemas de menor densidade (100 cidades) e quando utilizamos um menor número de grupamentos. As tabelas a seguir podem esclarecer melhor este fato. As duas primeiras mostram o número de vezes que cada heurística obteve o melhor resultado (menor ciclo) para cada função de limitação de pontos por grupamento (que também define o número de grupamentos da redução por dispersão) e para cada problema. A terceira e quarta contém a média percentual dos ganhos obtidos pela redução por dispersão em relação à redução convencional por cada função de redução e por problema respectivamente.

| Número de ocorrências de melhores resultados por função | | |
|---|---|---|
| Função | Redução Convencional | Redução por Dispersão |
| $3\sqrt{n}$ | 9 | 11 |
| $2\sqrt{n}$ | 11 | 9 |
| $3\frac{\sqrt{n}}{2}$ | 10 | 10 |
| $\sqrt{n}$ | 7 | 13 |
| $\frac{\sqrt{n}}{2}$ | 3 | 17 |
| $\frac{\sqrt{n}}{3}$ | 1 | 19 |
| $\frac{\sqrt{n}}{4}$ | 1 | 18 |
| $\frac{\sqrt{n}}{8}$ | 2 | 15 |
| $\frac{\sqrt{n}}{4}$ *** | 2 | 18 |
| $\frac{\sqrt{n}}{8}$ *** | 5 | 15 |

| Número de ocorrências de melhores resultados p/instância | | |
|---|---|---|
| Problema | Redução Convencional | Redução por Dispersão |
| C100A | 2 | 8 |
| C100B | 7 | 3 |
| C100C | 5 | 5 |
| C100D | 5 | 4 |
| C200A | 2 | 8 |
| C300A | 1 | 9 |
| C318A | 1 | 9 |
| C400A | 2 | 8 |
| C500A | 1 | 9 |
| C1000A | 1 | 9 |
| C1000B | 0 | 10 |
| C1000C | 0 | 10 |
| C1500A | 4 | 6 |
| C2000A | 0 | 10 |
| C2500A | 3 | 7 |
| C3000A | 1 | 9 |
| C4000A | 2 | 8 |
| C5000A | 3 | 7 |
| C8000A | 3 | 6 |
| C10000A | 1 | 7 |

| Vantagem da redução por dispersão sobre a convencional por função | |
|---|---|
| Função | Percentual de ganho médio |
| $3\sqrt{n}$ | - 0.70 |
| $2\sqrt{n}$ | - 0.36 |
| $3\frac{\sqrt{n}}{2}$ | + 0.44 |
| $\sqrt{n}$ | + 0.67 |
| $\frac{\sqrt{n}}{2}$ | + 2.70 |
| $\frac{\sqrt{n}}{3}$ | + 3.26 |
| $\frac{\sqrt{n}}{4}$ | + 3.41 |
| $\frac{\sqrt{n}}{8}$ | + 2.94 |
| $\frac{\sqrt{n}}{4}$ *** | + 3.09 |
| $\frac{\sqrt{n}}{8}$ *** | + 3.08 |

| Vantagem da redução por dispersão sobre a convencional por instância | |
|---|---|
| Problema | Percentual de ganho médio |
| C100A | + 1.82 |
| C100B | - 1:02 |
| C100C | + 0.08 |
| C100D | - 0.71 |
| C200A | + 2.55 |
| C300A | + 2.28 |
| C318A | + 3.07 |
| C400A | + 2.95 |
| C500A | + 2.58 |
| C1000A | + 2.80 |
| C1000B | + 3.23 |
| C1000C | + 1.84 |
| C1500A | + 2.90 |
| C2000A | + 2.36 |
| C2500A | + 1.45 |
| C3000A | + 2.40 |
| C4000A | + 1.85 |
| C5000A | + 1.97 |
| C8000A | + 0.86 |
| C10000A | + 1.14 |

## *IV.4 - A qualidade das soluções*

Nesta seção, nosso objetivo é tentar responder uma pergunta quase impossível: a que distância do valor ótimo se encontram as soluções fornecidas pela heurística proposta? É claro que teríamos uma imensa dificuldade para tentar calcular a solução ótima até mesmo para as instâncias de 100 cidades. Uma saída para este problema seria tentar estabelecer um termo de comparação com outras heurísticas conhecidas ao invés de centrar nosso objetivo na comparação com o valor ótimo. Um dos mais simples e eficientes critérios neste caso é o que foi proposto por *Ong e Huang*[ 60] , que avaliaram estatisticamente o comportamento de algumas heurísticas clássicas aplicadas à instâncias de até 1000 cidades no quadrado de lado 1. Em termos gerais, a conjectura de Ong e Huang é baseada em que o valor esperado de uma heurística *H* para uma instância de *n* cidades pode ser aproximado por uma reta da seguinte forma:

$$(H(n)) = b\sqrt{n} + c \quad \text{onde} \quad \frac{c}{\sqrt{n}} \to 0 \text{ quando } n \to \infty$$

A tabela a seguir mostra os coeficientes calculados por Ong e Huang pelo método dos mínimos quadrados para algumas heurísticas tradicionais:

| Heurística | b | c | coef. det. $R^2$ |
|---|---|---|---|
| 3-ótimo | 0.7425 | 0.5501 | 0.9997 |
| economias | 0.7936 | 0.4650 | 0.9997 |
| inserção mais barata | 0.8704 | 0.4397 | 0.9996 |
| vizinho mais próximo | 0.8952 | 0.7429 | 0.9995 |

O coeficiente de determinação $R^2$ indica a qualidade do ajuste dos coeficientes *b* e *c* e é calculado da seguinte forma:

$$R^2 = \frac{\sum(\dot{Y}_i - \bar{Y})^2}{\sum(Y_i - \bar{Y})^2}$$ onde $Y_i$ é o comprimento real do tour da amostra *i*, $\dot{Y}_i$ é o valor estimado do comprimento obtido a partir da reta de regressão e $\bar{Y}$ é a média dos comprimentos de todas as amostras. Quanto mais próximo de 1 for o valor de $R^2$ , mais precisos e ajustados serão os valores de *b* e *c*.

É evidente que um menor valor de *b* determina a dominância assintótica de uma heurística sobre outra, ou seja, implica em resultados de qualidade superior para grandes instâncias. Pelos critérios de Ong e Huang, o número de instâncias de até 1000 cidades dos testes computacionais aqui realizados não é suficiente para permitir um cálculo preciso dos coeficientes da reta para as heurísticas de redução aqui propostas. Apesar disso, os coeficientes foram calculados para cada função de limitação de pontos por grupamento de ambas as heurísticas, para que se possa fazer uma

comparação entre elas e as heurísticas tradicionais. As tabelas a seguir mostram estes resultados:

| Redução Convencional - amostras até 1000 nós | | | |
|---|---|---|---|
| Função | b | c | Coef. de det. $R^2$ |
| $3\sqrt{n}$ | 0.8430 | 0.7084 | 0.9927 |
| $2\sqrt{n}$ | 0.8460 | 0.7123 | 0.9946 |
| $3\frac{\sqrt{n}}{2}$ | 0.8541 | 0.4863 | 0.9977 |
| $\sqrt{n}$ | 0.8534 | 0.3882 | 0.9960 |
| $\frac{\sqrt{n}}{2}$ | 0.8648 | 0.1287 | 0.9982 |
| $\frac{\sqrt{n}}{3}$ | 0.8480 | - 0.0626 | 0.9983 |
| $\frac{\sqrt{n}}{4}$ | 0.8386 | - 0.2387 | 0.9970 |
| $\frac{\sqrt{n}}{8}$ | 0.7837 | 0.0447 | 0.9987 |
| $\frac{\sqrt{n}}{4}$ *** | 0.8306 | 0.0801 | 0.9976 |
| $\frac{\sqrt{n}}{8}$ *** | 0.7870 | 0.0892 | 0.9987 |

| Redução por Dispersão - amostras até 1000 nós | | | |
|---|---|---|---|
| Função | b | c | Coef. de det. $R^2$ |
| $3\sqrt{n}$ | 0.8193 | 1.0870 | 0.9942 |
| $2\sqrt{n}$ | 0.8430 | 0.6926 | 0.9953 |
| $3\frac{\sqrt{n}}{2}$ | 0.8514 | 0.4577 | 0.9947 |
| $\sqrt{n}$ | 0.8573 | 0.2852 | 0.9974 |
| $\frac{\sqrt{n}}{2}$ | 0.8463 | - 0.1384 | 0.9990 |
| $\frac{\sqrt{n}}{3}$ | 0.7945 | 0.0991 | 0.9976 |
| $\frac{\sqrt{n}}{4}$ | 0.7825 | 0.0999 | 0.9975 |
| $\frac{\sqrt{n}}{8}$ | 0.7394 | 0.6066 | 0.9991 |
| $\frac{\sqrt{n}}{4}$ *** | 0.7838 | 0.2636 | 0.9969 |
| $\frac{\sqrt{n}}{8}$ *** | 0.7392 | 0.6572 | 0.9991 |

Os resultados confirmam a superioridade da redução por dispersão (como já foi visto na seção anterior), pois em 9 das 10 funções avaliadas ela domina assintoticamente a redução convencional. Nas funções $\frac{\sqrt{n}}{4}$, $\frac{\sqrt{n}}{4}$ ***, $\frac{\sqrt{n}}{8}$ e $\frac{\sqrt{n}}{8}$ *** esta dominância é bastante acentuada. Como já se poderia prever, em todos os casos os valores do coeficiente de determinação $R^2$ não são tão próximos de 1 quanto os calculados por Ong e Huang, porém são suficientemente próximos para podermos admitir que os resultados obtidos tenham razoável precisão. Todas as heurísticas

tradicionais avaliadas por Ong e Huang tem complexidade elevada ( $O(n^2)$ ou maior), pois a simples necessidade do cálculo do vetor de distâncias implica em uma complexidade $O(n^2)$ . Nas heurísticas de redução aqui propostas a complexidade é aproximadamente $O(k^{3.8})$ no caso em que são calculados $k$ tours pelos nós representativos, e, $O(k^{3.3})$ quando calcula-se apenas $\sqrt{k}$ tours. Logo, se escolhermos funções convenientes, será sempre possível trabalhar com complexidade menor que a das heurísticas tradicionais. Qualquer das funções testadas tanto na redução convencional quanto na por dispersão, domina assintoticamente tanto a heurística de vizinho mais próximo quanto a de inserção mais barata. Na redução por dispersão, a função que proporciona os menores custos computacionais, $3\sqrt{n}$ , tem comportamento não muito distante do observado na heurística das economias; o mesmo ocorrendo com as funções $\frac{\sqrt{n}}{4}$ e $\frac{\sqrt{n}}{4}$ ***. Ainda para esta redução, as funções $\frac{\sqrt{n}}{8}$ e $\frac{\sqrt{n}}{8}$ *** apresentam um comportamento que sugere até mesmo uma dominância assintótica sobre a heurística 3-ótimo (que é uma das que apresenta melhores resultados em termos de qualidade), embora a relativa imprecisão dos cálculos aqui realizados não permita uma afirmação categórica neste sentido. Em síntese, podemos concluir que para os problemas de grande porte, com as heurísticas de redução aqui descritas é possível obter resultados de qualidade equivalente e em alguns casos melhor, com custo inferior ao das heurísticas convencionais.

## *IV.5 - Instâncias de maior densidade*

Examinando os resultados obtidos pelas heurísticas para as instâncias entre 2.000 e 10.000 nós, podemos observar um certo afastamento destes em relação ao valor esperado calculado pelo critério de Ong e Huang

na seção anterior. Curiosamente, as funções que induzem um menor número de retângulos (ou grupamentos) como $3\sqrt{n}$ , $2\sqrt{n}$ e $\frac{3}{2}\sqrt{n}$ apresentam uma progressiva melhora em relação ao valor esperado a medida que cresce o número de nós; e, por outro lado, as funções que induzem um maior número de retângulos (principalmente $\frac{\sqrt{n}}{4}$ , $\frac{\sqrt{n}}{8}$ , $\frac{\sqrt{n}}{4}$ *** e $\frac{\sqrt{n}}{8}$ *** ) experimentam uma piora progressiva em relação ao valor esperado. Podemos concluir então que a escolha da função adequada a cada problema feita unicamente pelo critério de Ong e Huang pode levar a resultados diferentes do que se poderia prever. Entretanto este fato não é válido para contestar a conjectura pois ela foi estabelecida a partir do estudo de heurísticas convencionais que atuam sempre de modo uniforme, diferentemente das heurísticas de redução. De qualquer forma, um estudo mais apurado da conjectura que incluisse instâncias superiores a 1000 nós no cálculo dos coeficientes das heurísticas convencionais seria bastante interessante como forma de reforçar ou não a sua viabilidade.

## *IV.6 - Influência do número de retângulos ou grupamentos*

A função de limitação de pontos por retângulo, que também foi usada para determinar o número de grupamentos na redução por dispersão, tem grande influência tanto na qualidade das soluções quanto no tempo computacional dispendido. Entretanto não existe uma relação direta entre ambos os fatores, ou seja, nem sempre uma função que induz a um maior gasto de tempo computacional fornece soluções de melhor qualidade. Isto pode ser facilmente visualizado se observarmos os coeficientes das retas de regressão que representam o valor esperado das soluções fornecidas por

cada função que foram calculados na seção anterior. A demanda por CPU das funções obedece as seguintes relações:

$$3\sqrt{n} \leq 2\sqrt{n} \leq \frac{3}{2}\sqrt{n} \leq \sqrt{n} \leq \frac{\sqrt{n}}{2} \leq \frac{\sqrt{n}}{4} \leq \frac{\sqrt{n}}{8} \quad \text{e}$$

$$\frac{\sqrt{n}}{4}^{***} \leq \frac{\sqrt{n}}{8}^{****}$$

Se observarmos as tabelas da seção anterior, podemos notar que para o primeiro grupo de funções, a medida que avança o consumo de tempo computacional, há uma piora da qualidade esperada da solução até determinado ponto, a partir do qual a tendência se inverte e a qualidade esperada começa a melhorar. Na redução convencional o ponto de inflexão é $\frac{\sqrt{n}}{3}$ e na redução por dispersão é $\frac{\sqrt{n}}{2}$. Curiosamente, ao contrário do que se poderia esperar, vemos que nem sempre um maior esforço computacional implica em uma qualidade esperada melhor. Uma possível explicação talvez esteja relacionada com a inclusão de nós extras, que representam os grupamentos ou retângulos, efetuadas pelas heurísticas de redução. Quanto maior o número de nós extras, menor será o trabalho de inserção realizado após o cálculo do ciclo inicial e vice-versa. Isto introduz um certo antagonismo entre as duas fases das heurísticas e o que se observa é que para "derrotar" o desempenho obtido por um alto índice de inserções é necessário um número significativo de grupamentos ou retângulos. Isto pode ser notado ainda mais claramente nos problemas mais densos (5.000 nós ou maiores) nos quais um elevado índice de inserções proporcionado pela função $3\sqrt{n}$ apresenta um excelente desempenho (vide seção III.3). Desta forma, se o objetivo principal for um baixo custo computacional, a melhor escolha para ambas as heurísticas é a função $3\sqrt{n}$. Caso contrário as melhores escolhas estão nas funções $\frac{\sqrt{n}}{4}^{***}$ e $\frac{\sqrt{n}}{8}^{***}$, as quais embutem um custo mais elevado , porém fornecem resultados de qualidade

bem superior (exceto nas instâncias acima de 8.000 nós onde o ganho não é considerável e onde a relação custo/benefício da função $3\sqrt{n}$ é excepcional). Finalmente, um último ponto que deve ser ressaltado é a grande versatilidade do critério de avaliação estatística de Ong e Huang que além de fornecer um termo de comparação com o desempenho de outras heurísticas pode facilitar bastante a escolha da função adequada nas heurísticas de redução tendo em vista as necessidades do usuário.

# CAPÍTULO V

## Conclusões

O PCV é um problema clássico exaustivamente estudado na literatura e para o qual, devido a provável inexistência de um algoritmo que o solucione de forma ótima, já foram desenvolvidas inúmeras heurísticas, algumas delas com excelentes resultados. Entretanto, devido a diversas limitações, frequentemente elas não podem ser aplicadas a instâncias com elevado número de nós. Este trabalho procurou abordar as técnicas já utilizadas para tratar instâncias geométricas de grande porte, tais como redução de nós, transformação em subgrafo esparso e particionamento. Dentre estas, a mais simples, redução de nós, foi escolhida para ser objeto de duas implementações a que nos referimos por redução convencional e redução por dispersão. Ambas são baseadas em três procedimentos principais: redução do problema principal, solução heurística do problema reduzido e inserção dos pontos do problema original no tour obtido para o problema reduzido. As duas heurísticas de redução propostas diferem apenas na fase inicial, ou seja, na fase de redução do problema original. A redução convencional divide recursivamente o retângulo que contém os nós do problema original até que os retângulos resultantes tenham um número máximo de nós pré-estabelecido em seu interior. A redução por dispersão procura obter um determinado número de pontos geometricamente dispersos e, a partir deles, forma grupamentos com os quais se obtém o problema reduzido. Os resultados colhidos indicam que a técnica utilizada na redução tem grande influência na qualidade das soluções obtidas tendo-se observado uma nítida predominância da redução por dispersão sobre a convencional. Ambas as técnicas, se adequadamente utilizadas, mostraram ser aplicáveis a instâncias de até 10.000 nós, as quais as heurísticas tradicionais dificilmente poderiam ser aplicadas. Além disso, uma avaliação estatística da redução por dispersão revelou que sua performance em termos de qualidade pode ser

praticamente comparável ao da heurística 3-ótimo, uma das mais eficientes que se conhece. É bastante curioso que um problema aparentemente não relacionado com o PCV como a dispersão possa contribuir com eficiência na construção de uma heurística para ele. Tal fato é bastante animador e imediatamente indica duas linhas de pesquisa bastante interessantes: a primeira diz respeito à busca de novas técnicas de redução, pois do mesmo modo que a dispersão se mostrou uma técnica superior à convencional, é possível que novas técnicas de redução com performance ainda melhor possam ser encontradas. A segunda seria relacionada ao emprego da própria dispersão também para heurísticas de particionamento e confrontar seus resultados com as técnicas já propostas e descritas no capítulo II.

# Bibliografia

1 - DANTZIG,G.B., FULKERSON,S.M. e JOHNSON,S.M., Solution of large-scale traveling salesman problem - Oper. Res. 2, 393-410 (1954).

2 - FLOOD,M.M., The traveling salesman problem - Oper. Res. 4, 61-75 (1956).

3 - CROES,G.A., A method for solving traveling salesman problems - Oper. Res. 6, 791-812 (1958).

4 - DANTZIG,G.B., FULKERSON,S.M. e JOHNSON,S.M., On a linear programming, combinatorial approach to the traveling salesman problem - Oper. Res. 7, 58-66 (1959).

5 - BELLMORE,M. e NEMHAUSER,G.L., The traveling salesman problem : A survey - Oper. Res. 16, 538-558 (1968).

6 - HELD,M. e KARP,R., The traveling salesman problem and minimum spanning trees - Oper. Res. 18, 6, 1138-1162 (1970).

7 - CHRISTOFIDES,N., Bounds for the traveling salesman problem - Oper. Res. 20, 1044-1056 (1972).

8 - LIN,S. e KERNIGHAN, B.W., An effecitve heuristic algorithm for the traveling salesman problem - Oper. Res. 21, 498-516 (1973).

9 - GOLDEN,B.L., BODIN,L.D., DOYLE,T. e STEWART,W., Approximate traveling salesman algorithms - Oper. Res. 28, 694-711 (1980).

10 - CHRISTOFIDES,N. e EILON,S., Algorithms for large scale traveling salesman problems - Oper. Res. Quart 23, 511-518 (1972)

11 - LENSTRA,J.K. e RINNOOY KAN,A.H.G., Some simple application of the traveling salesman problem - Oper. Res. Quart 26, 4, 717-733 (1975).

12 - PARKER,R.G. e RARDIN,R.L., Guaranteed performance heuristics for the bottleneck traveling salesman problem - Oper. Res. Letters 2, 269-272 (1984).

13 - PADBERG,M.W. e RINALDI,G., Optimization of a 532 city symetric traveling salesman problem by branch and cut - Oper. Res. Letters 6, 1, 1-7 (1987).

14 - HELD,M. e KARP,R.M., A dynamic programming approach to the sequencing problems - Siam J. Appl. Math 10, 1, 196-210 (1962).

15 - KARG, R.L.e THOMPSON,G.L., A heuristic approach to solving traveling salesman problems, Manag. Sci. 10, 2, 225-248 (1964).

16 - CROWDER,H. e PADBERG,M.W., Solving large scale symetric traveling salesman problems to optimality, Manag. Sci. 26, 5, 495-509 (1980).

17 - LIN,S., Computer solutions of the traveling salesman problem - Bell System Tech. J. 44, 2245-2269 (1965).

18 - RAYMOND,T.C., Heuristic Algorithm for the traveling salesman problem - IBM J. Res. Develop. 13, 400-407 (1969).

19 - KROLAK,P.D., FELTS,W. e MARBLE,G., A man-machine approach toward solving the traveling salesman problem - Comm. ACM 14, 327-334 (1971).

20 - HELD, M. e KARP, R.L., The traveling salesman problem and minimum spanning trees part II - Mathematical Programming 1, 6-25 (1971).

21 - ROSENKRANTZ, D.J., STEARNS, R.E. e LEWIS, P.M., An analysis of several heuristics for the traveling salesman problem - Siam J. Comput. 6, 3, 563-581 (1977).

22 - STEWART,W.R. Jr., A computationally efficient heuristic for the traveling salesman problem - Proc 13 th Annual Meeting of S.E. TIMS, 75-85 (1977).

23 - PARKER,R.G. e RARDIN,R.L., The traveling salesman problem: an update of research - Naval Research Logistics Quarterly 30, 1, 69-96, John Wiley, New York (1983).

24 - PADBERG,M.W. e RINALDI,G., A branch and cut algorithm for the resolution of large scale symetric traveling salesman problems - Research Report, New York University (1989).

25 - CHRISTOFIDES, N., The traveling salesman problem,em: Combinatorial Optimization, John Wiley, New York, 131-149 (1979).

26 - PADBERG,M.W. e HONG,S., On the symetric traveling salesman problem: a computational study - Math. Programming Study 12, 78-107 (1980).

27 - PAPADIMITRIOU,C.H. e STEIGLITZ,K., Combinatorial Optimization: Algorithms and complexity, Prentice-Hall, New Jersey, 406-483 (1982).

28 - MELAMED,I.I., SERGEEV,S.I. e SIGAL,I.K., The traveling salesman problem: Theoretical issues, Tradução para o inglês de Avtomatika i Telemekhanika 9, 3-33, Moscou (1989).

29 - TUTTE,W.T., Graph Theory - Encyclopedia of mathematics and its applications, vol. 21, Addison-Wesley, Massachusets (1984).

30 - GAREY,M.R.e JOHNSON,D.S., Computers and Intractability: a guide to the theory of NP-completeness, Freeman, San Francisco (1979).

31 - HOFFMAN,A.J. e WOLFE,P., History. In: LAWLER,E.L. et alii, eds., The traveling salesman problem: a guided tour of combinatorial optimization, John Wiley, Chichester, 1-15 (1985).

32 - SWARCFITER,J.L., Grafos e algoritmos computacionais - Campus, Rio de Janeiro (1983).

33 - BERGE,C., The theory of graphs and its applications - John Wiley, New York (1962)..

34 - BERGE,C., Graphs and hypergraphs - Nort-Holland, Amsterdam (1973).

35 - HARARY,F., Graph theory - Addison-Wesley, Reading (1969).

36 - GOLUMBIC,M.C., Algorithmic graph theory and perfect graphs - Academic Press, New York (1980).

37 - BIGGS,N.L., LLOYD,E.K. e WILSON,R.J., Graph theory 1736-1936 - Clarendon Press, Oxford (1976).

38 - REINELT,G., Fast heuristics for large geometric traveling salesman problems, Report n. 185, Institut fur Mathematik, Universitat Augsburg (1989).

39 - ERKUT,E., The discrete p-dispersion problem - European J. of Oper. Res. 46, 48-60 (1989).

40 - EASTMAN,W.L., Linnear programming with pattern constraints - Ph. D. thesis, Harvard University, Cambridge, MA (1958).

41 - LAWLER,E.L., LENSTRA,J.K., RINNOOY KAN,A.H.G. e SHNOYS,D.B., eds., The traveling salesman problem: a guided tour of combinatorial optimization, John Wiley, Chichester (1985).

42 - CHRISTOFIDES,N., Worst case analysis of a new heuristic for the traveling salesman problem - Report 388, Graduate school of industrial administration, Carnegie-Mellon University, Pittsburgh, PA (1976).

43 - CHRISTOFIDES,N., Vehicle Routing. In: LAWLER et alii, eds., The traveling salesman problem: a guided tour of combinatorial optimization, John Wiley, Chichester, 431-448 (1985).

44 - CLARKE,G. e WRIGHT,J.W., Scheduling of vehicles from a central depot to a number of delivery points - Oper. Res. 12, 568-581 (1964).

45 - GILLET,B.E. e MILLER,L.R., A heuristic algorithm for the vehicle dispatch problem - Oper. Res. 22, 340-349 (1974).

46 - CHISTOFIDES,N., MINGOZZI,A. e TOTH,P., The vehicle routing problem. In CHRISTOFIDES et alii, eds., Combinatorial optimization, John Wiley, Chichester, 315-338 (1979).

47 - FISHER,M.L. e JAIKUMAR,R., A generalized assignment heuristic for vehicle routing - Networks 11, 109-124 (1981).

48 - QUERIDO,T.M., Um estudo algébrico do problema do caixeiro viajante como um caso particular do problema quadrático de alocação - Tese de mestrado, IME, Rio de Janeiro (1987).

49 - CAMPELLO,R.E. e MACULAN, N., Algoritmos e heurísticas: desenvolvimento e avaliação de performance, EDUFF, Niteroi (1994).

50 - GILMORE,P.C., LAWLER,E.L. e SHMOYS,D.B., Well solved special cases. In: LAWLER,E.L. et alii, eds., The traveling salesman problem: a guided tour of combinatorial optimization, John Wiley, Chichester, 87-143 (1985).

51 - BURKARD,R.E. e VAN DER VEEN,J.A.A., Universal conditions for algebraic travelling salesman problems to be efficiently solvable - Optimization 22, 5, 787-814 (1991).

52 - KARP,R.M., Probabilistic analysis of partitioning algorithms for the traveling salesman in the plane - Math. Oper. Res. 2, 209-224 (1977).

53 - COOK,S.A., On the complexity of theorem-proving procedures - Proc. 3rd. Annual ACM Symp. Theory of Computing, 151-158 (1971).

54 - KARP,R.M., Reducibility among combinatorial problems. In: MILLER,R.E. e TATCHER,J.W., eds., Complexity of computer computations, Plenum Press, New York, 85-103 (1972).

55 - HOPFIELD,J.J. e TANK,D.W., Neural computations of decisions in optimization problems - Biol. Cybernetics 52, 141-152 (1985).

56 - DURBIN,R. e WILLSHAW,D., An analogue approach to the traveling salesman problem using an elastic net method - Nature 326, 689-691 (1987).

57 - OHYA,T., IRI,M. e MUROTA,K., Improvements of the incremental method for the Voronoi diagram with computational comparison of various algorithms - Journal of the Oper. Res. Soc. of Japan 25, 306-337, 1984.

59 - BALL,G.H. e HALL,D.J., Promenade - An on-line pattern recognition system - Rep. RADC-TR-67-310, AD 822174, Stanford Res. Inst., Menlo Park, California (1967).

60 - ONG,H.L. e HUANG,H.C., Asymptotic expected performance of some TSP heuristics: an empirical evalutation - European J. of Oper. Res. 43, 231-238 (1989).

61 - ARTHUR,J.L. e FRENDEWEY,J. O., Generating traveling salesman problems with known optimal tours - Research Report, Oregon State University (1985).

62 - BODIN,L., GOLDEN,B., ASSAD,A. e BALL,M., Routing and scheduling of vehicles and crews: the state of the art - Computers and Oper. Res., special issue vol. 10 no. 2 (1983).