Contribuições para Otimização em Grafos e Problemas de Percurso de Veículos: Sistema SisGRAFO

Marcos José Negreiros Gomes

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 OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Nelson Maculan Filho, D.Habil.
(presidente)

Prof. Arlindo Gomes de Alvarenga, D.Sc.

Prof. Eduardo Saliby, Ph.D

Prof. Luiz Satoru Ochi, D.Sc.

Prof. Rubem Pinto Mondaini, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
DEZEMBRO DE 1996

GOMES, MARCOS JOSÉ NEGREIROS

Contribuições para Otimização em Grafos e Problemas de Percursos de Veículos: Sistema SisGRAFO [Rio de Janeiro] 1996
VIII, 294 pgs., 29.7 cm, (COPPE/UFRJ, D. Sc., ENGENHARIA DE SISTEMAS E COMPUTAÇÃO, 1996)
TESE – Universidade Federal do Rio de Janeiro, COPPE
1 – Otimização em Grafos 2 – Modelagem Baseada em Grafos 3 – Roteamento de Veículos
I. COPPE/UFRJ II. Título(Série).

A Ana Cristina e Marina,
a meus Pais e Irmãos, com amor,
e ao Senhor, meu fiel e sábio amigo ...

Agradecimentos

Durante o curso de meu doutoramento e no projeto e implementação do sistema SisGRAFO, muitas pessoas me apoiaram com muito entusiasmo. O fato de serem pessoas tão amadas e respeitadas, para mim além de um dever é um enorme prazer registrar a minha gratidão.

Ao professor Nelson Maculan, por ter sempre acreditado no meu trabalho e disponibilizado tantas de suas preciosas horas dominiciais. Pela amizade marcante, pela orientação como pesquisador e como pessoa humana. Por sempre ter corrigido com suave rigor minhas falhas e deslizes. Por me suportar nos deslizes e estar presente nas alegrias e tristezas no curso deste doutorado... Esta tese tem a sua marca, pois foi feita dentro daquilo que pude aprender sobre sua visão e supervisão em oito anos e meio de trabalhos em conjunto.

Esta tese também tem um grande incentivador, e talvez uma das pessoas que mais me apoiaram em todos os momentos. Mesmo naqueles onde pensei em desistir, um contato era o bastante para prosseguir. Ao meu Irmão e Amigo, Francisco José Negreiros Gomes eis aqui o resultado de seu crédito, de seu apoio, e de sua amizade.

A minha esposa, que me deu Marina logo no início do doutorado, por ter se mantido firme ao meu lado com seu apoio, amor, paciência e amizade na longa e sofrida espera de conclusão deste trabalho.

A Marina, que chegou num momento difícil, porém me trouxe a possibilidade de aprender a vencer muitos obstáculos, aprender a ser pai, saber o que é amar profundamente e a estar sempre disposto a tentar concluir este trabalho.

A meu pai, Miguel Arcanjo, que jamais deixou de me alertar da importância deste trabalho. Que me apoiou sempre nos meus estudos, e que sempre facilitou as coisas nas horas mais difíceis.

A minha mãe e irmãos, que souberam sempre compreender a dificuldade de se formalizar um trabalho como este, e sempre me apoiaram dentro de suas possibilidades.

Aos amigos Max Padilha, Júnia Ângelo, Paulo e Amélia Pacheco, pelo grande apoio dado e pela calorosa amizade dispensada.

Aos amigos Xisto Medeiros, Cláudio e Ivan Ary, por terem aberto

tantas oportunidades e mostrado uma nova visão das pessoas que fazem o Ceará, e o grande apoio sempre prestado a mim.

Aos professores Christopher Jones e David Ronen, que apesar de não conhecê-los pessoalmente, ajudaram-me muito nas idéias e conclusões sobre as contribuições deste trabalho, via *e-mail*.

Ao amigo e professor Guy Barroso Silva, por me ter aberto as portas da Computação como professor da Universidade Estadual do Ceará, e pela calorosa e duradoura amizade dos idos e inesquecíveis tempos de Santo Inácio.

A Francisco Zampirolli, pelo trabalho de construção da interface e do editor de desenhos do SisGRAFO, e pela amizade dispensada.

Aos meus orientandos e orientados de Iniciação Científica da UECE, pois me mantiveram sempre motivado para a pesquisa em Ciência da Computação. Também aos colegas professores e alunos da UECE.

Aos colegas e alunos da Universidade Federal do Ceará, do departamento de Engenharia Mecânica, onde tive o prazer de lecionar.

Aos colegas da Universidade do Espírito Santo (UFES), do departamento de Informática.

Aos professores Luiz Satoru Ochi e Paulo Roberto Oliveira, pelo apoio nas qualificações do doutorado.

Aos professores Dina Cleiman, Jano Moreira, Ana Regina, Antônio Oliveira e Adilson Xavier, pelo apoio nos momentos difíceis do início de meu doutorado.

Aos que fazem a COPPE Sistemas e a UFRJ,

A CAPES, ao CNPq e à FUNCAP pelo suporte financeiro.

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

Contribuições para Otimização em Grafos e Problemas de Percursos de Veículos: Sistema SisGRAFO.

Marcos José Negreiros Gomes
Dezembro de 1996

Orientador: Nelson Maculan Filho
Programa: Engenharia de Sistemas e Computação

Esta tese apresenta o sistema SisGRAFO como uma ferramenta de Modelagem Visual Interativa que trata problemas de Otimização em Grafos. Discute-se as propriedades deste sistema também como um Sistema de Apoio à Decisão, e mostra-se detalhes de sua implementação computacional. A seguir discute-se os Problemas de Percurso de Veículos em Vértices e/ou Arcos, considerando-se o estado-da-arte das formulações, algoritmos exatos e heurísticos e extensões a problemas especiais. Trata-se também de novas estratégias de solução (heurísticas) para problemas Generalizados do Caixeiro Viajante (Problema de Compras), de Percurso de Veículos (Padrão e Multi- Compartimentados). O problema de Percurso em Arcos é explorado usando transformações originais de arcos em nós que permitem encontrar resultados aproximados de boa qualidade interessantes para o Carteiro Chinês, Carteiro Rural, *Stacker-Crane* e o Problema em Arcos com Demanda em redes gerais.

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

Contributions to Graph Optimisation and Vehicle Routing Problems:
SisGRAFO System.

Marcos José Negreiros Gomes
December, 1996

Thesis Supervisor: Nelson Maculan Filho
Department: Programa de Engenharia de Sistemas e Computação

This thesis shows the SisGRAFO system as a Visual Interactive Modeling tool that is prepared to work for Graph Optimisation Problems. Some properties of this system are discussed which also characterize it as a Decision Support System. A detailed description of the SisGRAFO implementation is also shown. The Vehicle Routing Problems on nodes and/or arcs are considered, where the state-of-the-art formulations, exact and heuristic methods are explored and some of their variants. Some new heuristic methods are considered to a Traveling Salesman Problem generalization (Shopping Problem) and to the General and Multi-Compartment Vehicle Routing Problem. The Routing Problem on Arcs are explored by original arc to node transformations which allow find good approximative results to the Chinese Postman, Rural Postman, Stacker-Crane and Capacitated Arc Routing Problems on general networks.

Índice

I Introdução **1**

I.1 Terminologia de Grafos Adotada 8

I.2 Organização da Tese 9

II Modelagem Visual com Grafos: Sistema SisGRAFO **12**

II.1 Modelagem Visual Interativa 13

II.1.1 Estrutura para Suporte à Decisão 14

II.1.2 Sistemas de Apoio à Decisão 17

II.1.3 Otimização e Visualização 20

II.2 Sistemas Baseados em Grafos 28

II.2.1 Gramática de Grafos 30

II.2.2 Formulação Baseada em Modelagem Interativa 37

II.2.3 Construção de Algoritmos em Modo de Edição 40

II.3 O Sistema SisGRAFO 41

II.3.1 Organização e Arquitetura 42

II.3.2 Editor de Grafos e Textos 54

II.3.3 Visualização das Soluções 57

II.3.4 Ambiente Computacional, Programação e Limitações 61

II.4 Principais Problemas Enfocados 62

II.4.1 Perspectivas do SisGRAFO 72

III O Problema do Caixeiro Viajante **74**

III.1 Definição do PCV 77

III.2 Formulações Matemáticas 78

III.2.1 Formulação Baseada em Atribuição 80

III.2.2 Formulação Baseada em Fluxo 81

III.2.3 Formulação em Fluxo com Multi-Comodidades 82

III.2.4 Formulação em Fluxo com Duas-Comodidades 83

III.3 Algoritmos para o PCV 84

III.3.1 Algoritmos Exatos 85

III.3.2 Algoritmos Heurísticos Padrões 87

IV Generalizações do Problema do Caixeiro Viajante 99

IV.1 O Problema de Compras 103

IV.1.1 Formulações do Problema de Compras 103

IV.1.2 Um Algoritmo Dinâmico de Compras 107

IV.1.3 Exemplo do Algoritmo Dinâmico 110

IV.1.4 Avaliação Computacional 113

V O Problema de Percurso de Veículos em Vértices 119

V.1 O PPV Genérico 122

V.1.1 Algoritmos Exatos do PPV Genérico 124

V.1.2 Principais Heurísticas do PPV Genérico 137

V.2 O PPV com Múltiplos Depósitos 146

V.3 O PPV com Janelas de Tempo 150

V.4 O PPV com Repuxo de Carga 153

V.5 O PPV com Estoque e o PPV Periódico 158

V.6 O PPV com Composição de Frota 162

V.7 O PPV com Demanda Dividida 164

V.8 O PPV com Múltiplos Compartimentos 165

VI O Problema de Percurso em Arcos **176**

VI.1 O Problema do Carteiro Chinês 186

VI.1.1 O PCC Não Orientado 188

VI.1.2 O PCC Orientado 191

VI.1.3 O PCC Misto 193

VI.1.4 O PCC Íngreme 199

VI.1.5 O PCC Hierárquico 202

VI.2 O Problema do Carteiro Rural 203

VI.2.1 O PCR Não Orientado 204

VI.2.2 O PCR Orientado 208

VI.2.3 O Problema *Stacker Crane* 208

VI.3 O Problema de Roteamento em Arcos com Demanda (PRAD) 212

VI.3.1 Formulações do PRAD 213

VI.3.2 Encontrando Limites Inferiores para o PRAD 216

VI.3.3 Métodos Heurísticos para o PRAD 219

VII Transformações de Arcos em Nós **226**

VII.1 Transformação de Pearn, Assad & Golden 227

VII.2 Transformação de Ulosoy 228

VII.3 Transformação por Quebra Simples de Ligações (QSL) 234

VII.4 Transformação Linha 245

VII.4.1 Definições e Teoremas Básicos para Transformação Linha . . . 245

VII.4.2 Matriz de Adjacências Valorada de $\mathcal{L}$ 251

VII.4.3 Aplicações da Transformação Linha 252

VII.4.4 Alguns Resultados Experimentais 254

VII.5 Minimizando o Esforço no Percurso em Ligações 258

VIII Conclusão **268**

Lista de Figuras

I.1 O problema de Percurso sobre Vértices. 4

I.2 O problema de Percurso sobre Ligações. 5

I.3 O problema de Percurso sobre Vértices e/ou Ligações. 6

II.1 Modelagem Algébrica adaptativa do PCV em LINGO. 23

II.2 Representação NETFORM, em Grafo, de um PPL Inteiro. 25

II.3 Representação NETFORM de um modelo de Localização Capacitado. 26

II.4 Representação LPFORM do modelo de Transporte. 26

II.5 Processo sistêmico em um SMBG. 29

II.6 Reroteando o cliente *a*. 32

II.7 Produção (simplificada) para rerotear um cliente. 32

II.8 Produção *RerotearCliente*. 33

II.9 Produção *Adicionarrota*. 34

II.10 Produção *Retirarcliente*. 34

II.11 Produção *Inserircliente*. 35

II.12 Representação regular baseada no Problema de Transporte. 36

II.13 Janela mostrando uma organização taxonômica de problemas no SisGRAFO. 43

II.14 Exemplo de Grafo Euclidiano. 45

II.15 Exemplo de Grafo Simétrico. 45

II.16 Exemplo de Grafo Misto. 46

II.17 Variação numa solução de caixeiro viajante euclidiano e misto, para um mesmo grafo base. 48

II.18 Código Pascal da estrutura do SisGRAFO. 49

II.19 Exemplo da formação da estrutura do SisGRAFO. 50

II.20 Representação mais detalhada da estrutura de arquivos do SisGRAFO. 50

II.21 Representação de armazenamento em arquivo texto de um grafo DLU.GRF do tipo misto produzido pelo SisGRAFO. 55

II.22 Ambiente de edição de figuras e grafos do SisGRAFO. 56

II.23 Exemplo de uma saída ambígua para um problema caminhos iguais. . 58

II.24 Impressão gráfica do resultado e *relatório* explicativo da solução numa situação de ambiguidade. 59

II.25 Impressão gráfica do resultado e *relatório* explicativo das soluções obtidas. 60

II.26 Exemplo de aplicação de Árvore Geradora Mínima. 63

II.27 Exemplo de aplicação (animada) de Fluxo em Redes de Custo Mínimo. 64

II.28 Situação anterior de transferência e proposta de melhoria do sistema de coleta de lixo de Fortaleza usando o algoritmo DLOC implementado no SisGRAFO. 66

II.29 Exemplo de Solução do 1-Caixeiro 68

II.30 Exemplo de Solução do m-Caixeiro. 68

II.31 Exemplo de Solução do Problema de Percurso de Veículos. 69

II.32 Exemplo de Solução (rota) de uma instância do Problema de Percurso Geral. 70

II.33 Exemplo de Solução de uma instância do Problema de Agrupamento Capacitado. 71

II.34 Diagrama das perspectivas do SisGRAFO. 74

III.1 Resequenciamento no posicionamento dos padrões para redução de perdas em um rolo de papel de parede. 76

III.2 Uma atribuição ótima de ϕ e uma Rota Piramidal obtida de ϕ. 79

III.3 Formação de subciclos em um problema de atribuição simples. 80

III.4 Uma solução heurística com curva de preenchimento. 88

III.5 Travessia ótima de um grafo usando Árvore Geradora Mínima. 89

III.6 Travessia ótima de um grafo usando Árvore Geradora Mínima e o Matching de Custo Mínimo. 90

III.7 Trocas 2-Otimais no caso simétrico. 92

III.8 Trocas Or-Otimais no caso simétrico. 94

III.9 Trocas 2-Otimais no caso Assimétrico. 95

IV.1 Formulação Dinâmica do Problema do Viajante de Compras. 107

IV.2 Uma solução do PC em um grafo Euclidiano. 116

IV.3 Uma solução do PC em grafo Simétrico. 117

IV.4 Uma solução do PC em grafo Misto. 117

V.1 Situação geográfica de distribuição para Clientes e suas Demandas. . 120

V.2 Uma solução do PPV aplicado à figura anterior. 120

V.3 Juntando clientes a rotas usando economias. 138

V.4 Proposta de Negreiros para melhoria do Algoritmo de Economias. . . 139

V.5 Situação geral de um PPV-MD 146

V.6 Problema de Despacho/Coleta de Malotes. 156

V.7 Situação de Despacho de Malotes. 157

V.8 Situação de Coleta de Malotes. 157

V.9 Situação geográfica para um PPV-MC. 171

V.10 Solução 1-Compartimento, $K = 5$ e $c = 3$. 173

V.11 Solução m-Compartimento, $K = 5$ e $c = 3$. 173

V.12 Situação geográfica para um PPV-MC real. 174

VI.1 Vista das sete pontes de Königsberg na ilha de Kneiphof sobre o rio Pregel. 176

VI.2 Situação de coleta definida para uma dada sub-região. 180

VI.3 Exemplo do percurso de um veículo *Fumacê*. 184

VI.4 Satisfação das Condições de Balanceamento em um grafo misto. . . . 187

VI.5 Um grafo aumentante do PCC. 188

VI.6 Grafo original orientado. 192

VI.7 Grafo orientado preparado para o Problema de Fluxo de Custo Mínimo. 192

VI.8 Grafo aumentado Euleriano resultante. 193

VI.9 Conversão de um elo em pseudo-ramos não unicursais. 196

VI.10 Grafo onde a aplicação direta do algoritmo de fluxo de custo mínimo atinge a solução do problema sem a formação de *circuitos triangulares*, custo da rota ótima=73. 197

VI.11 Grafo onde a aplicação direta do algoritmo de fluxo de custo mínimo não atinge a solução do problema sem a formação de *circuitos triangulares*, o fluxo triangular acontece em (6,32,31,6), ao redirecioná-lo a rota ótima é logo atingida com custo 92. 197

VI.12 Conversão de um elo em pseudo-ramos não unicursais para o PCC Íngreme. 202

VI.13 Grafo modificado para o PCR com G' conexo. 205

VI.14 Grafo relativo a um PSC. 209

VI.15 Transformação de um grafo G em um grafo H. 217

VII.1 Transformação de elos com demanda em nós, pela inclusão na rede de $3|R|$ nós em um grafo. 227

VII.2 Revisão da proposta de Pearn, Assad & Golden no caso de um elo sem demanda que não pertence a ciclos do grafo. 228

VII.3 Rede de serviço onde $q_{ij} \propto ij$ para todo $ij \in L$. 231

VII.4 Rede euleriana mínima aplicando-se 1-*Matching*. 232

VII.5 Transformação de Ulosoy. 232

VII.6 Situação de coleta de lixo domiciliar sobre ruas e pontos especiais no bairro da URCA, Rio de Janeiro. 235

VII.7 Quebra da rede proposta inicialmente por Negreiros 90. 238

VII.8 Quebra da rede (em elos) proposta por Negreiros & Maculan 92. . . . 239

VII.9 Problema exemplo proposto por Pearn 91. 243

VII.10 Quebra da rede - Transformação de arcos em nós. 244

VII.11 Exemplos de Transformação Linha. 246

VII.12 Casos de *Caminhos Adjacentes* na formação de um Grafo Linha. . . 246

VII.13 Uma transformação de G_d para um digrafo linha $\mathcal{L}_d$. 247

VII.14 Formação linha f-conexa de um grafo f-conexo. 248

VII.15 Composição de um percurso equivalente. 249

VII.16 Composição de um circuito equivalente. 250

VII.17 Composição de um caminho-circuito. 250

VII.18 Exemplo de um circuito hamiltoniano linha e circuito euleriano após composição. 250

VII.19 Grafo linha de um grafo não orientado. 251

VII.20 Mudança proposta à TL para o problema do Carteiro Íngreme. . . . 254

VII.21 Avaliando subciclos. 259

VII.22 Grafo Euleriano para avaliação de rota com esforço mínimo. 260

VII.23 Juntando subciclos gerados. 262

VII.24 Representação de movimentação por transformação linha. 264

VII.25 Grafo exemplo de um circuito euleriano direcionado. 265

VII.26 Grafo T_G referente à Transformação Linha. 266

Lista de Tabelas

II.1 Estrutura de apoio à Tomada de Decisão. 16

III.1 Representação de Algoritmos Genéticos em Matemática. 96

IV.1 Custos ($) das distâncias entre os centros de consumo. 111

IV.2 Preços e estoques nos centros de compras 111

IV.3 Análise do comportamento do Algoritmo Dinâmico para o PC. 115

V.1 Resultados obtidos com a proposta de melhoria. 140

V.2 Coordenadas e demandas por produto por cliente. 172

V.3 Custo por unidade de produto. 172

V.4 Impostos (%) por unidade de produto. 172

V.5 Comparação entre as soluções obtidas do exemplo. 172

V.6 Comparação entre as soluções obtidas em instâncias com dados reais de demanda, custos de produto e transporte, e posicionamento de clientes. 175

VII.1 Descrição das estratégias usadas no algoritmo MARCOLIX 242

VII.2 Comparação entre as Transformações - PCC. 255

VII.3 Quadro de atribuição de serviço. 261

VII.4 Quadro de atribuição de serviço. 262

VII.5 Limites angulares correspondentes à movimentação de veículos entre ligações sucessivas da rota euleriana. 265

VII.6 Adjacência e custos dos arcos de T_G. 266

Capítulo I

Introdução

A preocupação com o desenvolvimento de Sistemas de Apoio à Decisão (SADs) robustos e cada vez mais integrados a especialistas e usuários é uma façanha que muitos autores e corporações ao redor do mundo tentam atingir. Este intento pode ser difícil de ser alcançado por uma única pessoa, porém com todo o esforço que se tem feito nos últimos anos na área, é possível que muitos especialistas reunidos consigam lograr este feito que será sempre um fato marcante na história da computação.

Sistemas de Apoio vêm avançando de forma especializada em diversas áreas. Dos diversos tipos de sistema de apoio existentes pode-se citar os: Baseados em Redes Neurais, de Simulação, de Árvores de Decisão, de Planejamento e Aprazamento, de Distribuição e Transporte, de Controle de Tráfego, de Programação Matemática, etc. Todos estão em geral voltados para aplicações específicas ou genéricas com a intenção de formar um quadro de decisão que ajude ao usuário no encaminhamento eficaz de respostas a suas perguntas.

Este trabalho surgiu de minha necessidade de conhecimento sobre a arte da Programação Matemática (PM), mais especificamente na Otimização em Grafos dirigida à Modelagem Visual Interativa. Por oito anos trabalhei com insistência em problemas da área em geral ligados a aplicações reais complexas que envolvem muitas informações associadas ao contexto de análise dos problemas. Neste tempo pude sentir por diversas vezes a dificuldade que tem o especialista e o usuário de Programação Matemática na visualização dos problemas complexos.

Trabalhei dentro deste período no projeto de concepção e geração de um sistema de apoio à tomada de decisões baseado em Grafos, sistema SisGRAFO. Nas primeiras linhas do SisGRAFO, a simples visão de que era possível posicionar vértices, e movê-los na tela de um microcomputador, foi para mim o motivo da concepção de como seria seu funcionamento. Vi daí como os problemas que

estudava poderiam ser facilmente escritos em uma linguagem estruturada usando uma representação de modelos baseada em Grafos.

Uma vez vencida a etapa de representação (geração de um TAD Grafo), a etapa de organização estrutural do sistema foi se configurando à medida que implementava alguns dos diversos algoritmos hoje inseridos. Novas restrições, visualizações mais confortáveis, recursos de impressão e textos, enfim, grande parte das idéias propostas por [Jones 90] foram incluidas no sistema.

Os quatro primeiros dos cinco anos de desenvolvimento do SisGRAFO foram realizados penosamente em um microcomputador PC-386 SX 16MHz com 2Mb de memória RAM, e nos últimos meses do último ano quase que concomitante a esta tese um modelo PC 100MHz com 16Mb (Pentium-Aptiva IBM) aliviou toda a exaustiva fase de testes e análises de grande parte das contribuições que serão colocadas a seguir neste texto.

À medida que os algoritmos mostravam-se robustos, e que conseguia manter intacto o propósito do sistema, pelas dificuldades de gerenciamento de memória, espaço em disco rígido, dentre outras, muitos resultados interessantes apareciam vindo exclusivamente da interação com os problemas e da visualização dos resultados. Várias publicações conjuntas e individuais foram então possíveis, trabalhos e algumas consultorias a orgãos públicos e privados foram prestadas com o uso do SisGRAFO. Tudo isto contribuiu para que esta tese fosse elaborada enfocando os principais problemas estudados ao longo dos cinco anos de seu desenvolvimento.

O SisGRAFO é pois uma ferramenta baseada em Modelagem Visual Interativa (MVI), uma técnica de Sistemas de Apoio ao Gerenciamento (SAGs) ligada ao apoio à decisão através da representação visual de problemas. Seu objetivo é de auxiliar usuários na compreensão e resolução de problemas relacionados à Otimização em Grafos, e seu principal instrumento de modelagem visual são grafos, ou um conjunto formado por vértices (ou pontos) e ligações (entre os pontos).

Conforme [Jones 90], a representação visual de vários problemas combinatórios por grafos é tão robusta quanto a sua representação em linguagem conceitual. Para tornar mais fácil a manipulação de grafos, Jones considera que através de Gramáticas de Grafos se pode ter um elemento de troca das informações sobre diversos parâmetros de uma instância de um problema ou na estruturação de um problema sobre grafos.

A fase de visualização das instâncias e das soluções servem como um elemento de análise e interação entre o sistema e o usuário, sobre um contexto de tomada de decisão. Elas estão bem definidas em [Jones 94] como elementos indispensáveis ao ciclo de vida do processo de construção de um modelo.

Alguns sistemas como o SisGRAFO estão em pleno desenvolvimento,

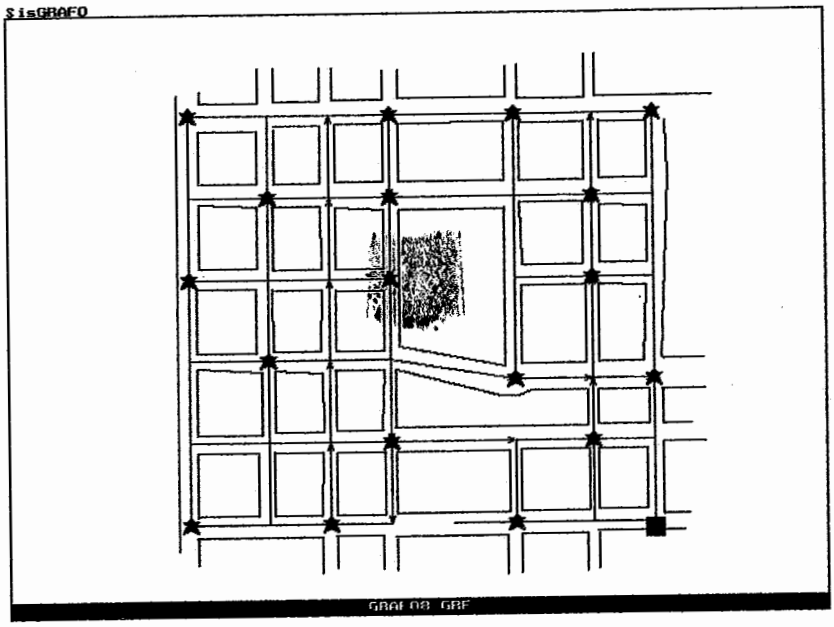
NETWORKS, NETWK, NETPAD, dentre outros, usando recursos de grafos. Outros já estão comercialmente disponíveis, tais como: ARCVIEW, MAPINFO, ROADMAP, TRANSCAD, dentre outros, na linha de Sistemas de Informações Geográficas (SIGs) que usam grafos no tratamento de alguns problemas de decisão. A proposta básica dos sistemas baseados em grafos é o uso de modelagem de problemas através de grafos. Isto inclui a formulação do problema básico, a manipulação de instâncias, e a contextualização do problema em uma situação de planejamento e tomada de decisão. Nos SIGs esta abordagem não faz parte do sistema, é em geral incorporada através do uso de linguagens interpretadas que chamam algoritmos externos (em linguagem de máquina ou em C) ao sistema cujo contexto é em geral formado por mapas de situações, referenciados usualmente pelas coordenadas cilíndricas da terra.

O SisGRAFO incorpora problemas básicos e bem resolvidos tais como: Árvores Geradoras, Fluxo em Redes, Problema de Transporte, Fluxo de Custo Mínimo, Caminhos Mínimos, Atribuição Simples e de 1-*matching* valorado. O sistema também incorpora problemas de larga escala cujas soluções são obtidas através de heurísticas: Problemas do Caixeiro Viajante, do Carteiro Rural, de Roteamento Geral, de Localização de Depósitos, de Agrupamento Capacitado, dentre outros. Como um sistema de grafos, a sua formalização e a gama de problemas atendidos não encontra similar na literatura, constituindo-se ele mesmo uma contribuição em PO.

Do ponto de vista dos problemas de percursos, considera-se três enfoques de análise: Percursos somente sobre vértices de um grafo, somente sobre ligações e em vértices e/ou ligações. O primeiro caso se refere aos problemas ligados ao Problema do Caixeiro Viajante, onde são dadas cidades, lojas ou pontos geograficamente dispersos e um viajante partindo de um deles percorrerá os demais apenas uma vez retornando para a origem com a menor distância total de percurso, figura I.1. O segundo caso se refere aos problemas do Carteiro Rural onde são dados segmentos (trechos) de ruas de uma cidade e um carteiro partindo do posto central irá percorrê-las na menor distância total, figura I.2. No terceiro caso tem-se o problema de Percurso Geral, onde a partir de um posto de serviço um carteiro deverá visitar lojas (vértices) e/ou segmentos de rua (quarteirões), com o objetivo de perfazer o menor percurso no atendimento a estes elementos, figura I.3.

Grande parte do trabalho desenvolvido envolveu os problemas do caixeiro viajante e do carteiro chinês, esta tese considera também as contribuições dadas nestes problemas específicos de Otimização em Grafos, colocando de forma suscinta o seu desenvolvimento e algumas de suas particularidades no encaminhamento de soluções.

Nos Problemas do Caixeiro Viajante, desenvolve-se aqui uma análise de funcionamento de algoritmos heurísticos de melhoria de rotas (r-otimal). Já bastante estabelecidos e divulgados pela literatura. Em extensões do problema, no



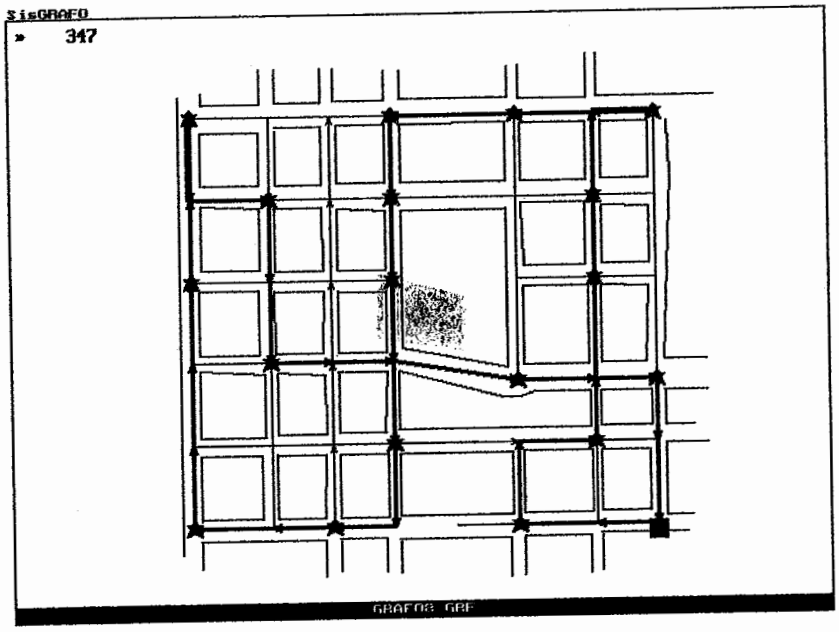
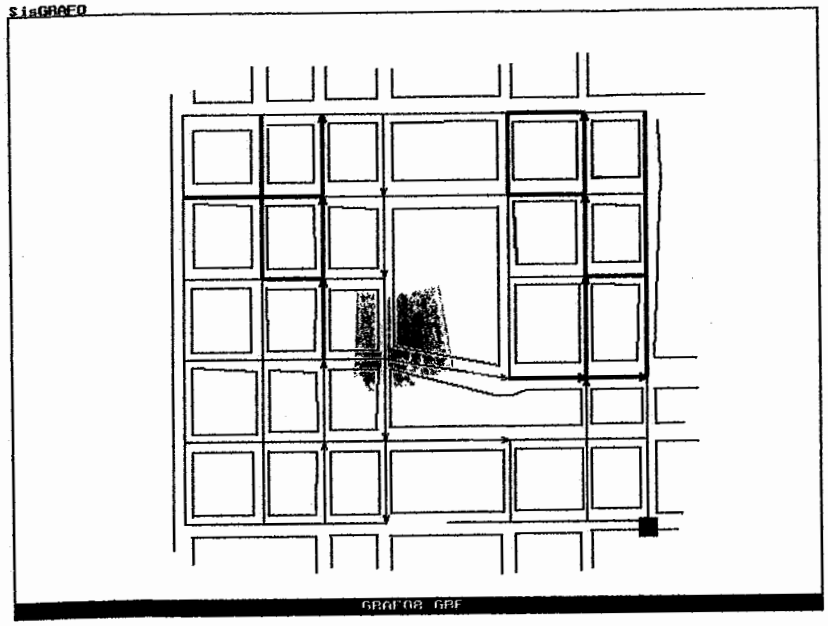


Figura I.1: O problema de Percurso sobre Vértices.



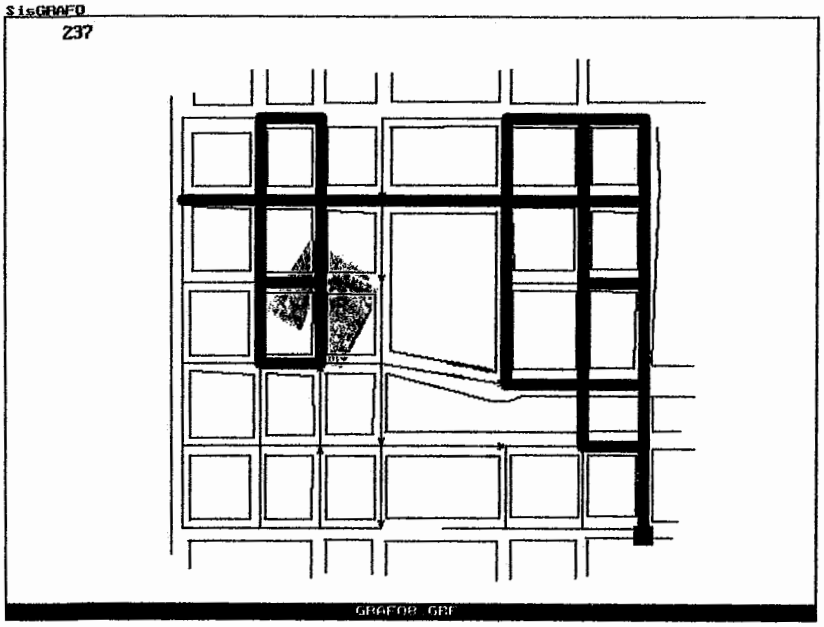
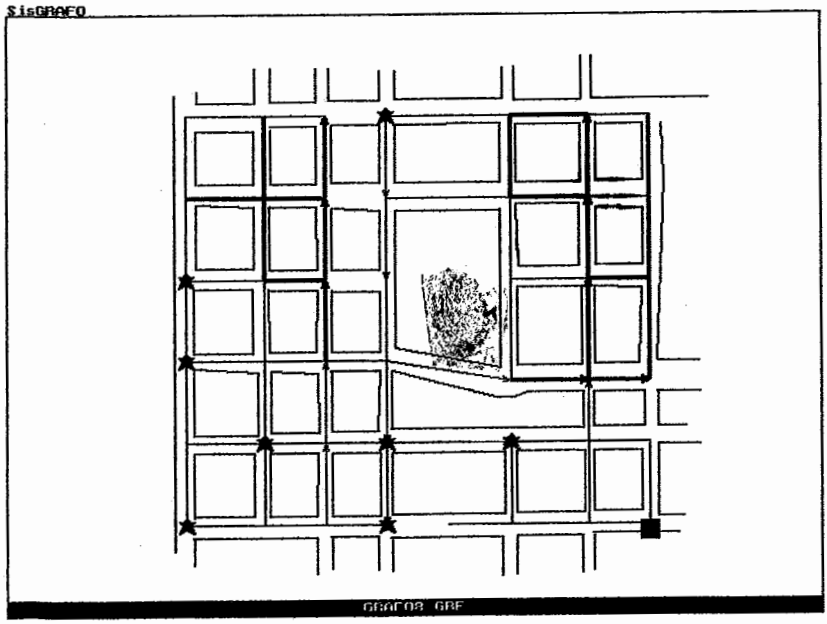


Figura I.2: O problema de Percurso sobre Ligações.



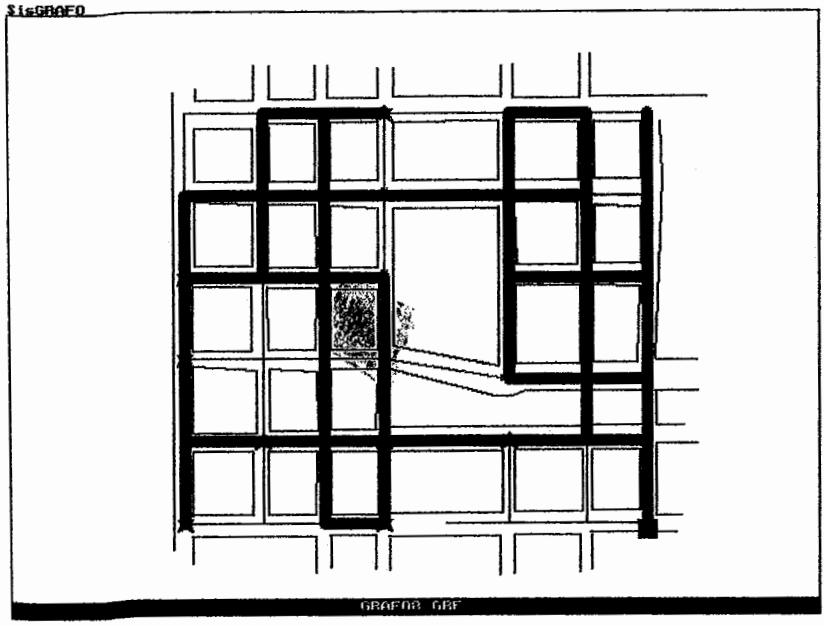


Figura I.3: O problema de Percurso sobre Vértices e/ou Ligações.

caso o problema do Caixeiro Restrito, propõe-se uma nova abordagem, mais geral ao Problema do Caixeiro de Compras (*Traveling Purchaser Problem*), proposto por [Ramesh 78], ou o Problema de Compras, usando um algoritmo dinâmico que busca soluções próximas do ótimo em iterações polinomiais.

O Problema de Percursos de Veículos em Nós também surge dentro de uma abrangência, onde se coloca principalmente o problema original padrão proposto por [Dantzig & Ramser 59], em seguida explora-se as principais técnicas (exatas e heurísticas) que estão sendo tratadas pela literatura no seu desenvolvimento e aplicação. Neste caso, propôs-se um aumento na eficácia de obtenção de soluções com algum trabalho adicional sobre a formação de rotas, no algoritmo de economias proposto por [Clarke & Write 64].

Dentro dos diversos casos especiais do Problema de Percurso de Veículos em Nós, o Problema com Múltiplos Compartimentos foi explorado com o objetivo de tratar a possibilidade de atendimento de uma ou mais de uma demanda por um mesmo compartimento utilizando uma metodologia heurística, baseada numa modificação do método de economias. A aplicação surge na entrega de combustíveis a postos, ou no roteamento de navios graneleiros.

Nos Problemas de Percurso em Arcos, explora-se algumas das suas diversas aplicações a várias atividades do dia-a-dia, no que concerne os serviços a domicílio. Explora-se muito as formulações e os principais algoritmos exatos relacionados ao problema do carteiro chinês e suas variações. Coloca-se uma aplicação do algoritmo proposto por [Sherafat 88] para o Problema do Carteiro Chinês Íngreme. No caso dos problemas relacionados ao Carteiro Rural, explora-se as formulações, os limites inferiores e alguns dos principais algoritmos ligados ao problema, os quais usam em geral algoritmos de árvores geradoras (floresta geradoras), de 1-*matching*, de caminho mínimo e de fluxo em redes.

Por fim, explora-se o problema de percursos em arcos e em nós, ou o Problema Geral de Percursos, através de idéias de transformações em redes. Aborda-se o problema só de arcos através de uma proposta simples de transformação, transformação linha, que pode tratar estes problemas e muitos outros ligados a arcos. Para o Problema Geral de Percursos, mostra-se uma idéia de quebra de rede proposta inicialmente por [Pearn et al 87] e modificada por [Negreiros & Maculan 92a]. Os resultados mostraram-se interessantes do ponto de vista prático, sobre instâncias extraidas diretamente da literatura.

Todas as aplicações aqui apresentadas foram desenvolvidas diretamente usando o sistema SisGRAFO, cujo ambiente mostra-se favorável à análise de modelos de otimização em grafos. Explorou-se bastante os recursos visuais de modo a permitir ao usuário uma maior compreensão dos resultados e soluções, foram incluidos até recursos de animação ao sistema com o objetivo de prestar uma maior

compreensão dos problemas do modelo de percurso de veículos, e ao problema de fluxo em redes. Uma interface adaptativa também foi construida de modo a permitir o desenvolvimento da ferramenta para aplicações específicas (módulo SApplication).

O SisGRAFO é então um ambiente que favorece um estreitamento entre problemas práticos de combinatória e a sua compreensão por parte dos mais diversos usuários e especialistas da área.

I.1 Terminologia de Grafos Adotada

Uma consideração sobre alguns conceitos fundamentais da teoria de grafos, tal que seja facilitada a compreensão deste texto, faz-se necessária.

Um grafo G é um conjunto formado por uma dupla de conjuntos, Vértices (V) e de Ligações entre os vértices (L), definido por $G = (V, L)$. O conjunto formado pelas ligações é formado por uma dupla de conjuntos (E, A), onde E é o conjunto das ligações não orientadas (sentido de travessia não definido), e A é aquele formado pelas orientadas (sentido de travessia definido).

Chama-se de grafo simétrico ou não-orientado de $G = (V, E)$, onde $V \neq \emptyset$ é o conjunto de vértices ou nós e $E \neq \emptyset$ é o conjunto de elos (ligações sem orientação). Elos paralelos são todos aqueles que possuem seus vértices extremos distintos iguais. A notação de um elo pode ser colocada como ij ou (i, j), onde os nós $i, j \in V$. Denota-se o conjunto de elos incidentes a i por $g(i)$, e sua cardinalidade por $d_G(i)$, que representa o grau do vértice i no grafo G. Um vértice de grau par é aquele que possui cardinalidade $d_G(i)$ par, e um vértice de grau ímpar, possui cardinalidade $d_G(i)$ ímpar.

Digrafos ou grafos orientados, $D = (V, A)$, onde $V \neq \emptyset$ é o conjunto de vértices ou nós e $A \neq \emptyset$ o conjunto de arcos. Chama-se um arco $a \in A$ incidente de i e incidente para j por $a =< i, j >$. Para um nó $i \in V$, denota-se o conjunto de arcos incidentes de i (incidentes para i) por $g^+(i)$ ($g^-(i)$), e sua cardinalidade denotada por $d^+(i)$ ($d^-(i)$), o número de arcos que possuem como *Rabo* (*Cabeça*) o vértice i.

Grafos mistos, $G = (V, E, A)$, onde $V \neq \emptyset$ é o conjunto de vértices ou nós, E o conjunto de elos e A o conjunto de arcos. Num grafo misto, o grau de um vértice i, será a soma do número de elos incidentes com i, o número de arcos incidentes de i, e o número de arcos incidentes para i.

Seja $G = (V, E)$ um grafo e $u, v \in V$. Uma sequência $\omega =< i_0 i_1,\ i_1 i_2,\ \ldots,\ i_{k-1} i_k >$ de elos de E com $i_0 = u$ e $i_k = v$ é chamado um *Percurso* entre u e v. ω é *Aberto* se $u \neq v$ e *Fechado*, caso contrário. ω é um *Caminho* se $i_s \neq i_t$,

para $s \neq t$, $s \geq 0$, $t \leq k$. Se ω é um *Caminho* e $i_k i_0$ e E, então a sequência $< i_0 i_1,\ i_1 i_2, \ldots,\ i_{k-1} i_k,\ i_k i_0 >$ é um *Ciclo*. Se um *Percurso* existe entre quaisquer pares de vértices de V, então G é dito ser *Conexo*. Um percurso de G contendo cada elo de E exatamente uma vez é chamado de *Percurso de Euler*. E um grafo que possui um *Percurso de Euler*, é um *Grafo Euleriano*.

Seja $D = (V, A)$ um digrafo, e u e v dois nós em V. Uma sequência $\omega =< (i_0, i_1),\ (i_1, i_2),\ \ldots,\ (i_{k-1}, i_k) >$ de arcos em A com $i_0 = u$ e $i_k = v$ é um *Percurso Direcionado* de u para v. ω é *Aberto* se $u \neq v$ e *Fechado* caso contrário; e ω é um *Caminho Orientado* se $i_s \neq i_t$ para $s \neq t$, $s \geq 0, t \leq k$. Se ω é um *Caminho Orientado* e $(i_k, i_0) \in A$, então a sequência $< (i_0, i_1), (i_1, i_2),\ \ldots\ ,\ (i_{k-1}, i_k) >$ é um *Ciclo Orientado*. Com qualquer digrafo $D = (V, A)$, pode-se associar o grafo $G(V, E)$, onde $E = ij \mid (i, j) \in A$. Diz-se que D é *Conexo* se G é *Conexo*. Um *Percurso Orientado Fechado* em D contendo cada arco de A exatamente uma vez é chamado um *Percurso Euleriano Orientado* e um digrafo D que possui um *Percurso Euleriano Orientado* é um *Digrafo Euleriano*.

Uma árvore T é um grafo acíclico, onde o número de arestas é igual ao número de vértices menos um. Uma árvore expandida T é uma árvore de G, onde todos os vértices de G são atingidos. Uma árvore é orientada quando a partir de uma raiz são emanadas todas as ligações entre vértices. Ou seja, há sempre um caminho direto entre a raiz e qualquer vértice da árvore, e dos pais para seus descendentes, o contrário não se aplica.

Todo grafo $G' = (V', L')$, onde $V' \subseteq V$, $L' \subseteq L$, pode ser definido como um subgrafo de $G = (V, L)$, ou seja: $G' \subseteq G$.

Um grafo G é fortemente conexo (f-conexo) quando para todo vértice existe um caminho ligando a qualquer outro vértice. Uma componente conexa, é um subgrafo f-conexo de G.

Um grafo é atribuido G_a quando seus vértices e ligações recebem valores referentes a custos, pesos, ou qualquer outro termo que defina valor ou estado sobre o elemento. Um Grafo de Atribuição é quando se permite a um grafo assumir um conjunto de valores ou estados de forma pré-definida.

I.2 Organização da Tese

Este texto está organizado como segue.

No capítulo II discute-se sobre Modelagem Visual Interativa e o Sistema SisGRAFO, onde busca-se colocar o desenvolvimento conceitual e de aplicações de Sistemas de Apoio à Decisão, o Problema da Otimização e Visualização os quais

servem como elementos básicos na formação do SisGRAFO. Por fim mostra-se aspectos gerais sobre a estrutura básica de desenvolvimento do sistema e seus algoritmos.

O Problema do Caixeiro Viajante é apresentado a seguir, no capítulo III, onde discute-se as aplicações, as formulações principais da literatura, alguns dos métodos exatos e heurísticos desenvolvidos. No capítulo IV, tem-se a análise de variações do Problema do Caixeiro Viajante, o Caixeiro Restrito, onde considera-se as diversas aplicações de cada problema especial. Uma atenção é dada aos problemas de Compras e do Caixeiro Coletor de Valores.

O Problema de Percurso de Veículos em Vértices, é tratado de forma suscinta no capítulo V, desde sua formulação, alguns métodos exatos e alguns dos principais métodos heurísticos propostos pela literatura. Modelos com restrições adicionais e específicos de aplicações reais são mostrados como derivações do problema principal, e alguns dos métodos de solução são abordados. Uma atenção maior é dada ao problema de percurso de veículos com múltiplos compartimentos, considerando uma formulação elástica de particionamento de conjuntos retirada da literatura e uma metodologia heurística baseada no método de economias com algumas adaptações especiais.

Os problemas relacionados ao Percurso em Arcos, advindos do Problema do Carteiro Chinês são tratados no capítulo VI. Enfoca-se a princípio os problemas do ponto de vista das aplicações a ele relacionadas, a seguir explora-se o contexto das formulações do problema e dos diversos algoritmos propostos para todas as suas variações consideradas pela literatura.

No capítulo VII, considera-se o Problema Geral de Percursos através do uso de transformações de arcos em nós com técnicas de transformação linha e de quebra de arcos em vértices-arcos rotulados, as quais juntas atingem resultados interessantes e de boa qualidade a instâncias do Carteiro Chinês: Não-Direcionado, Direcionado, Misto, *Stacker-Crane*, do Carteiro Rural, e do Problema Capacitado em Arcos.

Capítulo II

Modelagem Visual com Grafos: Sistema SisGRAFO

As aplicações (softwares) de computadores relacionadas ao suporte ao gerenciamento e principalmente na tomada de decisão, têm tido nos últimos anos um crescimento singular. Um número de ferramentas de escritório (processadores de texto, planilhas eletrônicas, e atualmente redes globais) reduziram bastante a dificuldade no tratamento e armazenamento de informações gerenciais, as quais hoje estão sendo usadas com o intuito de facilitar a tomada de decisão gerencial e operacional da grande maioria das empresas mundiais.

Com o processo de globalização da economia, a eficácia nos processos supera a espectativa sobre a eficiência, tornando-se um elemento indispensável à sobrevivência das empresas face à competição acirrada do mercado. A metodologia para atingí-la se torna cada vez mais acessível em função das disponibilidades e ferramentas computacionais existentes e em franco desenvolvimento, e de novas técnicas de Pesquisa Operacional (PO) no tratamento de problemas antes dificilmente implementáveis.

A grande dificuldade reside ainda na representação fiel dos problemas de gerenciamento. O conceito de representação fiel, vem da necessidade de apresentação de um modelo, o qual deve ser factível de ser implementado e compreendido em sua totalidade. No início, em PO, esta idéia foi relegada à expressividade de um conjunto de variáveis sobre a formação dos custos e no encaminhamento da função objetivo. Hoje, porém, a possibilidade de retratação da realidade através de novas técnicas mais aproximadas à compreensão do homem comum, tem garantido um impulso sem precedentes à Pesquisa & Desenvolvimento (P&D) de sistemas para auxílio à tomada de decisão.

Sistemas de Apoio à Decisão (SAD), Sistemas de Apoio à Decisão

Gerencial (SADG), Sistemas de Informações Executivas (SIE), Sistemas Especialistas (SE), e Redes Neurais Artificiais (RNA) são algumas das tecnologias que foram desenvolvidas as quais vêm mudando essa situação. Estas tecnologias são chamadas de Sistemas de Apoio ao Gerenciamento (SAG), [Turban 95].

Os SAGs tiveram um grande impulso a partir do momento em que a representação visual da realidade tornou-se uma ferramenta computacional disponível e interativa ao gerente, facilitando a compreensão dos modelos reais, aprimorando a qualidade das informações adquiridas, interagindo com várias soluções propostas sobre diferentes perspectivas, e implementando soluções. Mais importante, foi que a visualização permitiu que novas propostas de solução pudessem ser diretamente consideradas e novos procedimentos algoritmicos pudessem ser projetados. A visualização tornou-se enfim um instrumento simbólico de compreensão da realidade, e um recurso sem precedentes na história de sistemas computacionais.

Este capítulo trata basicamente dos elementos formadores de um SAG baseado em Modelagem Visual (MV) com Grafos, que se aplica a uma série de problemas de gerenciamento e operação relacionados ao dia-a-dia de planejadores, especialistas e gerentes. Inicialmente, na seção II.1, discute-se sobre Modelagem Visual Interativa (MVI), caracteriza-se SADs, e considera-se as vantagens (na otimização) da modelagem através da visualização. Na seção II.2 descreve-se conceitualmente o que são grafos, o que é gramática de grafos e o seu uso através de editores de instâncias e de esquemas, explorando-se também a possibilidade de se construir métodos de resolução de problemas através da junção de processos (algoritmos) ordenados em produções pré-definidas. Na seção II.3 descreve-se o **SisGRAFO**, sua formalização e estrutura, comparada aos modelos mais proeminentes, os aspectos de edição de instâncias universais (grafo), a interação com o usuário, os problemas tratados no sistema e as perspectivas futuras de implantação e projeto.

II.1 Modelagem Visual Interativa

Modelagem Visual Interativa (MVI) iniciou com Hurrion em 1970, cujo objetivo principal era de melhorar os registros de implementação nos estudos de PO. No início a maioria dos trabalhos foram feitos em simulação, porém ela tem se tornado uma ferramenta importante em várias áreas de PO.

MVI é um modelo baseado em computador que incorpora a representação visual do sistema que está sendo modelado, no qual os usuários podem fazer mudanças interativamente e logo podem ver os efeitos. Apesar de uma implementação bem sucedida de MVI ser mais do que a soma de suas partes, a combinação de uma representação visual efetiva e facilidades apropriadas para interação provêem os elementos básicos para que o usuário aprenda sobre o problema e sobre

seus resultados, [Belton & Elder 91].

Os modelos visuais interativos representam um passo esperado no desenvolvimento da PO. No início, os primeiros pesquisadores de PO desenvolviam seus modelos simbólicos de problemas os quais eram posteriormente resolvidos à mão, muitas vezes por diversas pessoas. Os computadores eram usados inicialmente como máquinas de calcular. Com o reconhecimento de que eles poderiam também resolver modelos de sistemas, surgiu a simulação por computador. Do cálculo de modelos de sistemas, progrediu-se para a modelagem de sistemas por computador. Com o estabelecimento de modelos interativos, a adição de gráficos seria um passo lógico na melhoria da comunicação entre o usuário e o modelo. A inevitável sequência de desenvolvimento sugeriu que um modelo audio-visual interativo não estaria longe, [Bell 91].

A MVI surgiu como uma técnica de PO. Isto foi um marco, uma vez que a experiência de mais de quarenta anos no uso de modelos no auxílio à tomada de decisão obteve uma apresentação mais agradável e superior no tratamento dos problemas gerenciais.

Os modelos visuais interativos são uma forma de SAG, porisso o grande interesse por parte dos especialistas da área no desenvolvimento de ferramentas MVI para SADs, SEs, SADGs, SEIs, etc.

II.1.1 Estrutura para Suporte à Decisão

Uma idéia geral sobre suporte à decisão foi sugerida em dois trabalhos, [Antony 65] e [Simon 77], conforme estrutura apresentada na tabela II.1. À esquerda da tabela, tem-se que o processo de tomada de decisão recai sobre decisões altamente estruturadas (programáveis) a altamente desestruturadas (não programáveis). Um processo estruturado se refere àqueles que são rotineiros e repetitivos para os quais soluções padrões existem. Já os não estruturados são *confusos*, problemas tão complexos que não possuem soluções que possam ser sistematizadas diretamente. Um problema não estruturado é aquele em que nenhuma das fases de Inteligência (busca por condições que requerem decisões), Projeto (invenção, desenvolvimento, e análise de cursos possíveis de ação) e Escolha (seleção de uma ação entre as disponíveis) são estruturadas. As decisões onde algumas, mas não todas, das fases são estruturadas são ditas semi-estruturadas.

Num problema estruturado, os procedimentos de obtenção da melhor solução (ou pelo menos uma solução razoavelmente boa) são conhecidos. Por exemplo, se um problema deseja encontrar o nível apropriado de estoque ou a decisão de uma estratégia ótima de investimento, os objetivos estão definidos de forma clara, o gerente pode pois usar modelos clássicos de solução. SAGs tais como SADs e SEs

são apropriados para estes casos.

Em problemas não estruturados, a intuição humana é frequentemente a base da tomada de decisão. Problemas tipicamente desestruturados incluem o planejamento de novos serviços a serem oferecidos, contratação de um executivo, ou a escolha de um portifólio de projetos para o próximo ano. Os problemas semi-estruturados e não estruturados, envolveriam uma combinação entre procedimentos analíticos padrões e julgamento individual.

A segunda parte das idéias gerais está baseada na taxonomia proposta por [Antony 65], que define três categorias que englobam todas as atividades gerenciais:

1. Planejamento Estratégico, objetivos de longo prazo e políticas para alocação de recursos;
2. Controle Gerencial, aquisição e utilização eficiente de recursos de acordo com os objetivos organizacionais;
3. Controle Operacional, A execução efetiva e eficiente de tarefas específicas.

Na última coluna à direita e última linha abaixo da tabela II.1, tem-se as tecnologias necessárias para o apoio às várias decisões não estruturadas e semi-estruturadas, onde as técnicas convencionais até então existentes de PO eram insuficientes. Nestes casos foi proposto o uso de um sistema de suporte à informação, SADs. SEs foram introduzidos alguns anos depois, sendo mais adequados a tarefas que requerem especialização.

As atividades mais estruturadas e de controle operacional são em geral executadas por gerentes iniciantes, enquanto tarefas que requerem a intuição e a experiência, são de responsabilidade dos executivos principais. Isto significa que SADs, SEIs, Redes Neurais e SEs são mais aplicáveis aos executivos principais e/ou a profissionais tarimbados em problemas específicos e complexos.

Dentre as diversas razões para o desenvolvimento de SAGs, algumas são específicas a certos sistemas, como segue abaixo, conforme [Turban 95]:

- SADs

  Quando na implantação de modelos matemáticos de PO e questões de julgamento pessoal sobre soluções ocorrem. Questões do tipo *O que ... se* poderiam ser respondidas com a introdução do elemento de análise de sensibilidade e alta interação com o usuário.

- SADGs

Tabela II.1: Estrutura de apoio à Tomada de Decisão.

Tipo de Decisão	Tipo de Controle			Suporte Necessário
	Operacional	Gerencial	Planejamento Estratégico	
ESTRUTURADA	- Contas a Receber; - Pedidos a Entrar.	- Análise de Lucro - Previsão a Curto Prazo - Relatórios pessoais.	- Gerenc. Fin. (Investimento), - Localização de Facilidades; - Sistemas de Distribuição;	- SAGs; - Modelos de PO; - Transação de Processos.
SEMI ESTRUTURADA	- Produção; - Aprazamento; - Controle de Estoques.	- Avaliação de Crédito; - Níveis de Lucro; - Layout de Fábricas; - Planej. de projetos;	- Construir novas fábricas; - Ligações e Aquisições; - Planej. de novos produtos; - Compensação; - Garantia da Qualidade.	SADs
NÃO ESTRUTURADOS	- Sel. da capa de uma revista; - Compra de um software; - Aprov. de empréstimo.	- Negociação; - Recrutar um executivo; - Compra de um hardware; - Fazer lobby.	- Planejamento de P&D; - Des. novas tecnologias; - Planejar respons. social;	- SADs, - SEs, - Redes Neurais.
Apoio Necessário	SAGs, PO	PO, SADs, SEs, SIEs.	SIEs, SEs Redes Neurais	

Muitas decisões em organizações são tomadas em grupo, de forma coletiva. A tarefa de juntar um grupo de pessoas é geralmente difícil e dispendiosa. Além disto, as reuniões de grupo tomam em geral um bom tempo e as decisões resultantes são frequentemente de baixa qualidade.

Tentativas de melhorar o trabalho de grupos são feitas com a aplicação de tecnologiais de *grupoware*, sistemas eletrônicos de reunião, tele-conferências, sistemas colaborativos, e sistemas de apoio à decisão em grupo.

- SIEs

São em geral desenvolvidos para:

- Servir as informações necessárias aos executivos;
- Prover uma interface extremamente amigável para o executivo;
- Atingir aos estilos de decisão individuiais dos executivos;
- Prover um acesso rápido à informação detalhada daquilo que está por trás dos textos, números, ou dos gráficos;
- Filtrar, comprimir e corrigir dados e informações;
- Identificar problemas (oportunidades).

- SEs

Quando uma empresa tem uma decisão complexa a tomar ou um problema a resolver, em geral ela chama especialistas para resolvê-lo. Estes especialistas têm conhecimento específico e experiência na área. São capazes de dar alternativas, prever as chances de sucesso, e os custos que ocorrerão no negócio. À medida que o problema é mais desestruturado, mais especializada é a consulta.

Um Sistema Especialista é um pacote de tomada de decisão ou um "resolvedor" de problemas em hardware ou em software, que pode atingir a um nível de

desempenho comparável a (ou sempre maior) do que um especialista humano em algum problema bastante específico.

Sistemas Especialistas são um ramo da Inteligência Artificial. Em geral estão associados a aplicações tais como diagnóstico médico, exploração mineral, e configuração de computadores. Eles também estão se expandindo em aplicações complexas tais como no gerenciamento de Ativo e Passivo de empresas, no planejamento de corporações, análise de falhas, etc.

- Redes Neurais

A aplicação de todas as tecnologias anteriores são baseadas no uso de dados explícitos, informação ou conhecimento, os quais são armazenados em um computador e manipulados como for necessário. Entretanto, no mundo real tem-se que considerar a possibilidade da ausência da informação, ou do conhecimento. As pessoas tomam muitas decisões que em geral estão baseadas em informações parciais, incompletas ou inexatas. Tais condições são criadas, por exemplo, em ambientes de rápida mudança. Tomadores de decisão estão usando suas experiências para considerar estas situações; ou seja, eles estão revendo suas experiências e aprendendo com elas para saber o quê fazer em situações similares onde réplicas exatas de informações ou conhecimentos não estão disponíveis.

Nos casos anteriores, o componente de aprendizado pelo computador não existe. Uma tecnologia que tenta chegar próximo a este limite é chamada de Computação Neural ou Redes Neurais Artificiais (RNAs). A tecnologia que emprega a técnica de reconhecimento de padrões, tem sido usada com sucesso em algumas aplicações comerciais e financeiras, no controle de doenças cardíacas e infantis, em previsões, etc.

II.1.2 Sistemas de Apoio à Decisão

A idéia de SADs iniciou com [Keen & Scott-Morton 78], que usaram o termo *Apoio à Decisão* para relacionar o uso de computadores a:

1. Assitência a gerentes no processo de decisão de tarefas semi-estruturadas;
2. Apoio, ao invés de substituição, no julgamento gerencial;
3. Melhoria na efetividade (eficácia) da tomada de decisão ao invés da eficiência.

Muitos autores mostraram-se controversos ao relacionamento considerado por Keen & Scott-Morton. Em geral, a literatura não tem uma definição

universalmente aceita. Como não há consenso sobre o que é um SAD, há pelo menos um acordo sobre o que se espera das características de um SAD.

As características ideais de um SAD são numeradas a seguir, porém não significa que todo SAD deva ter todas elas, a maioria dos SADs existentes incorporam apenas parte delas, [Turban 95].

1. Incorporar Decisões Semi-Estruturadas

   Os SADs provêem o suporte a tomadores de decisão principalmente em situações semi e não estruturadas juntando o julgamento humano e a informação computacional. Tais problemas não podem ser resolvidos por outros sistemas computacionais de gerenciamento nem tampouco por métodos e ferramentas padrões quantitativas.

2. Projetado para Gerentes de Níveis Diferentes

   O apoio é dado para diferentes níveis gerenciais, indo dos executivos principais aos gerentes de produção.

3. Projetado para Grupos ou Indivíduos

   O apoio é dado tanto a grupos quanto a indivíduos. Muitos problemas organizacionais envolvem decisões grupais. Os problemas pouco estruturados necessitam frequentemente o envolvimento de vários indivíduos de diferentes departamentos e níveis de organização.

4. Decisões Interdependentes ou Sequenciais

   SADs dão apoio a algumas decisões interdependentes e/ou sequenciais.

5. Apoio à Inteligência, Projeto e Escolha

   O apoio a todas as fases da tomada de decisão, inclusive na implantação.

6. Suporte a uma Variedade de Estilos de Decisão e Processos

   Adaptação aos atributos individuais de tomada de decisão de certos tomadores de decisão.

7. Adaptabilidade e Flexibilidade

   É adaptativo ao tempo. O tomador de decisão pode ser reativo, sendo capaz de confrontar condições de mudanças rápidas e adaptar o SAD para atingir estas mudanças. Os SADs são flexíveis quando os usuários podem adicionar, deletar, combinar, mudar e rearranjar elementos básicos (dando respostas rápidas a situações inesperadas). Esta capacidade torna possível a análise periódica.

8. Facilidade de Uso

   Os usuários devem se sentir à vontade com o sistema. O sistema deve ser amigável, flexível, com fortes capacidades gráficas, e de linguagem acessível o mais aproximado da conversação entre duas pessoas. Isto implica num modo interativo.

9. Efetividade (Eficácia) ao invés de Eficiência

   O SAD deve tentar melhorar as soluções de forma efetiva, ao invés de explorar sua eficiência (custo de tomar a decisão, incluindo as mudanças nos tempos computacionais).

10. Controle Humano da Máquina

    O tomador de decisão tem o completo controle sobre todos os passos no processo de resolução de um problema. Um SAD deve objetivar o apoio e não a substituição às decisões humanas. O tomador de decisão deve sobrepor às recomendações da máquina a qualquer instante.

11. Uso Evolutivo

    O SAD deve estar continuamente em processo de aprendizado, melhoria e desenvolvimento.

12. Fácil de Construir

    Usuários finais devem ser capazes de construir sistemas simples por eles mesmos. Sistemas maiores podem ser construidos em instituições onde os usuários são atendidos com uma tênue assistência dos especialistas da informação.

13. Modelagem

    Os SADs utilizam em geral modelos (padrões customizados) para a análise de situações de decisão. As capacidades de modelagem permitem a experimentação com diferentes estratégias sobre diferentes configurações. Tais experimentos podem chegar a novas idéias ou mesmo num aprendizado.

14. Conhecimento

    SADs avançados estão equipados com uma componente de conhecimento que permite uma solução eficiente e efetiva de problemas muito difíceis.

Como se pode ver, é muito difícil reunir todas as características acima descritas em um único SAD. A totalidade dos SADs existentes possuem apenas uma parte deste conjunto de características, as quais algumas são realmente relevantes e sem elas não há um mínimo SAD:

- Facilidade de uso e interatividade (Ferramenta orientada ao usuário);

- Modelagem e análise de sensibilidade;
- Possibilidade de decisões locais individualizadas;
- Interferência humana nas decisões;
- Facilidade de uso e de Construção;
- Desenvolvimento e uso evolutivo.

A diferença básica entre sistemas MVI e SADs está no processamento das transações, onde nos MVIs o elemento simbólico visual é tratado como a representação básica no interfaceamento das informações, [Bell 91]. É possível, porém, que alguém possa achar que as duas coisas se confundem.

II.1.3 Otimização e Visualização

Os objetivos da Visualização e PO são muito próximos, quando se trata de dar uma idéia do comportamento e solução de problemas complexos. PO usa algoritmos matemáticos enquanto a Visualização estuda a representação. As técnicas de PO em desenvolvimento, têm se respaldado em uma forte fundamentação teórica de matemática e ciência da computação. Pode-se provar normalmente que uma solução particular de um problema de programação matemática (PPM) é ótima, que um algoritmo irá convergir, ou que a solução, quando não for ótima, é muito próxima do ótimo. Provas matemáticas dos efeitos de visualização parecem menos promissoras. No melhor dos casos, pode-se conduzir experimentos bem projetados de modo a estabelecer a qualidade de diferentes formas de visualização, [Jones 94]

Alguns autores consideram que o estilo experimental de pesquisa não se encaixa na tradição da PO. No entanto, áreas de referência tais como psicologia cognitiva linguística, interação homem-computador, e lógica formal podem contribuir a teorias de PO bem desenvolvidas. Muitas destas disciplinas não têm sido consideradas tradicionalmente como parte das áreas de PO.

Ciclo de Vida de Preparação de um Projeto

No ciclo de vida de preparação de um projeto de programação matemática, alguns pontos são úteis e relevantes: Modelo conceitual, formulação, coleta de dados, execução de algoritmo, análise de solução e apresentação de resultados, [Jones 94].

1. <u>**Modelo Conceitual**</u>

O início de um processo de resolução de um problema envolve a tradução de objetivos e idéias mal-enfocadas em algo mais coerente, apesar de ainda imprecisas, tem-se daí a descrição do problema a ser atacado. A descrição do problema deve ser compreensível ao tomador de decisão, que em geral tem pouco conhecimento sobre o problema. As representações podem ser textos simples descrevendo o problema em linguagem natural ou através de diagramas formais ou informais. A MVI é considerada por muitos autores como uma metodologia geral para a criação de representações conceituais apropriadas, [Hurrion & Secker 78], [Hurrion 86], [Bell 85a]-[Bell 91].

Alguns autores acreditam que na fase de resolução de problemas deve-se concentrar fortemente o desenvolvimento de uma representação conceitual apropriada, com um atraso na formulação do modelo até que uma representação conceitual precisa (em geral gráfica) seja desenvolvida. A representação conceitual deve ser uma forma primária de definir o problema.

Bell considerou que o objetivo da técnica envolve a conciliação entre a representação conceitual com a representação matemática. Em alguns sistemas MVI (como o GENETIK[1], por exemplo) explora-se a idéia de desenhar figuras para a representação conceitual, e em seguida relacioná-las a algoritmos, ou a PPMs. A técnica MVI que permite esta facilidade pode ser vista como um movimento maior chamado de *Projeto Centrado no Usuário*, que se estende a áreas relacionadas a: fatores humanos, ergonometria, e desenho industrial.

Alguns autores propuseram linguagens de alto nível específicas para programação matemática, as quais representam problemas no plano conceitual. Em *Modelagem Estruturada*, Geoffrion propôs uma teoria de modelagem que provê algumas representações que tencionam representar modelos não apenas da otimização, mas de um largo espectro de técnicas usadas em PO, [Geoffrion 87] - [Geoffrion 89]. Baseado nas técnicas de Lógica de Predicado de Primeira-Ordem, [Krishnan 90] propôs uma linguagem de alto nível chamada PM para a especificação de problemas de programação linear. Uma representação gráfica da PM foi proposta por [Jones & D'Souza 92].

Basicamente as representações em MVI são menos abrangentes que a modelagem estruturada e a PM. Estas são mais abstratas e podem dar um maior sentimento ao especialista sobre a solução de um modelo. Porém são menos eficazes na apresentação e na compreensão, uma vez que toda a formulação conceitual do modelo depende de quem o fez.

2. **Formulação**

Uma vez que o modelo conceitual do problema foi construido, o próximo passo em programação matemática envolve a construção de uma representação formal matemática do problema, que é a formulação.

[1]INSIGHT International Ltda.

Dois conceitos surgem, o modelo genérico e uma instância do modelo. No modelo genérico se preconiza a formulação matemática do problema, variáveis, índices e inequações as quais estão relacionadas ao modelo genérico. As instâncias do modelo são situações referentes ao modelo genérico onde apenas as constantes e custos são atualizados no modelo. A maioria dos autores recomendam uma separação entre modelo e instâncias de modelo de modo a tornar reutilizável o modelo geral. A questão, porém, é como representar os modelos genéricos e ligá-los a suas instâncias.

Alguns estilos de linguagens de formulação foram propostos, os quais incluem as formulações:

Algébricas

Procuram duplicar a notação matemática. Caracterizam-se pela geração do modelo por linhas, uma vez que cada restrição ou conjunto delas pode ser especificado diretamente. Alguns dos softwares que adotam esta linguagem são: GAMS, AMPL, MPL, MODLER, LINGO e SML, figura II.1.

Bloco Estruturadas ou Orientada a Processo

Esta formulação organiza uma matriz de programação linear em blocos, onde cada bloco representa a interseção de um conjunto de variáveis de decisão e restrições. Elas representam modelos genéricos primariamente, e instâncias de modelos são representadas na forma de matrizes tabulares.

Linguagem Orientada a Objetos

A orientação a objetos permite que componentes atuais do modelo sejam reutilizados de forma controlada, organizando-os em hierarquia. Este estilo de organização parece natural para otimização uma vez que, por exemplo, um PPL pode ser considerado como um caso especial da Programação Não Linear, um problema de fluxo de custo mínimo é um caso especial do problema de Programação Linear.

Planilhas

As planilhas eletrônicas têm obtido um largo uso nos últimos anos na solução de problemas de programação matemática, uma vez que os softwares mundiais da área, LOTUS 1-2-3 e Microsoft EXCEL, incorporaram algoritmos de programação linear e não linear. Uma vez que uma planilha é construida, o usuário apenas indica a célula cujo valor será otimizado, as células cujos valores serão mudados (variáveis de decisão), e os limites a atribuir a outras células (restrições), e o algoritmo irá considerar o conjunto de variáveis de decisão e os valores para otimizar a célula indicada. Aqui, a formulação não precisa ser especificada como uma matriz de restrições, mas pode ser vista apenas como uma planilha.

Linguagens Gráficas Visuais

```
MODEL:
   1]  ! O Problema do Caixeiro Viajante para as cidades de
   2] Atlanta, Chicago, Cincinnati, Houston, LA, Montreal;
   3] SETS:
   4]  CITY / 1.. 6/: U; ! U( I) = sequencia no. de cidade;
   5]  LINK( CITY, CITY):
   6]       DIST,  ! A Matriz de Distancias;
   7]          X;  ! X( I, J) = 1 Se o elo I, J eh usado;
   8] ENDSETS
   9]DATA:   ! A matriz de distancias, nao precisa ser simetrica;
  10]  DIST =   0  702  454  842 2396 1196
  11]         702    0  324 1093 2136  764
  12]         454  324    0 1137 2180  798
  13]         842 1093 1137    0 1616 1857
  14]        2396 2136 2180 1616    0 2900
  15]        1196  764  798 1857 2900    0;
  16]ENDDATA
  17]
  18]! O Modelo: Ref. Desrochers & Laporte, OR Letters, Fev. 91;
  19]  N = @SIZE( CITY);
  20] MIN = @SUM( LINK: DIST * X);
  21]  @FOR( CITY( K):
  22]  !  Deve entrar nela;
  23]    @SUM( CITY( I)| I #NE# K: X( I, K)) = 1;
  24]  !  Deve sair dela;
  25]    @SUM( CITY( J)| J #NE# K: X( K, J)) = 1;
  26]   ! A forma fraca de restricoes de quebra de subrotas ilegais;
  27]   !  OBS: Elas nao sao muito fortes para problemas muito grandes;
  28]   @FOR( CITY( J)| J #GT# 1 #AND# J #NE# K:
  29]       U( J) > U( K) + X ( K, J) -
  30]       ( N - 2) * ( 1 - X( K, J)) +
  31]       ( N - 3) * X( J, K); );
  32]    );
  33] ! Fazer X 0/1;
  34] @FOR( LINK: @BIN( X););
  35] ! Para a primeira e ultima parada nos conhecemos...;
  36] @FOR( CITY( K)| K #GT# 1:
  37]      U( K) < N - 1 - ( N - 2) * X( 1, K);
  38]      U( K) > 1  + ( N - 2) * X( K, 1);   );
END
```

Figura II.1: Modelagem Algébrica adaptativa do PCV em LINGO.

Em Programação Matemática, a maioria das linguagens visuais estão relacionadas a grafos, como um modelo fundamental de construção. Na maioria destas linguagens, cada restrição ou variável de decisão representa um vértice ou um arco. Um estilo de representação chamado NETFORM foi proposto por [Glover et al 77], onde um problema qualquer de programação linear pode ser usado pela redução a uma representação de grafos, como um problema de fluxo (com ganhos) de custo mínimo em redes, figura II.2.

Simbologia

x_i	→	Variável x_i	k	→	Restrição (k)
c	→	Custo	g	→	Restrição (k)
#	→	Restrição (k)	(a, b)	→	Limites Assumidos
CF	→	Custo Fixo (não importa o fluxo que passa por ele)			

Modelo

$$
\begin{aligned}
&\text{Minimizar } 3x_1 + 4x_2 + 7x_3 + 5x_4 \\
&\quad \text{sujeito a:} \\
&\quad (1)\ 2x_1 + 4x_3 - 4x_4 \geq 2 \\
&(2)\ -x_1 + 10x_2 - 2x_3 - 6x_4 \leq 7 \\
&\quad (3)\ 4x_1 + x_2 + 4x_4 = 5 \\
&\quad (4)\ x_3 + x_4 \geq 1 \\
&\quad x_i \in \{0, 1\}
\end{aligned}
$$

Num modelo com custo fixo, por exemplo, tem-se um modelo de localização capacitado a seguir, que é representado por um grafo na figura II.3.

Supondo,

c_{ij}, o custo de envio de produto da fábrica i ao cliente j;

F_j, é o custo fixo de uso da fábrica j;

r_{ij}, é a demanda de produto do cliente i pela fábrica j;

b_j, é a oferta possível da fábrica j;

Representação

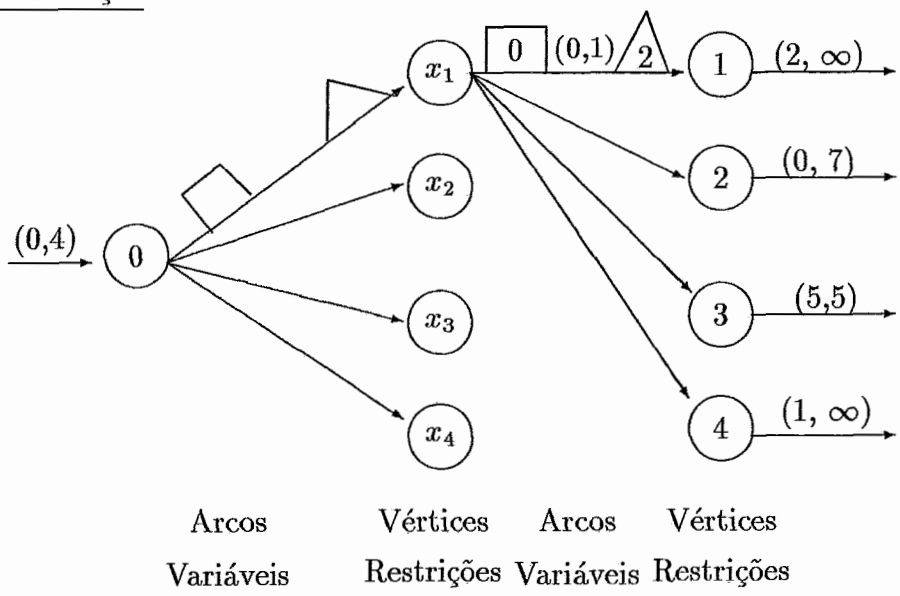


Figura II.2: Representação NETFORM, em Grafo, de um PPL Inteiro.

$$x_{ij} = \begin{cases} 1, & \text{se o cliente } i \text{ é servido pela fábrica } j \\ 0, & \text{caso contrário.} \end{cases}$$

$$Z_j = \begin{cases} 1, & \text{se a fábrica } j \text{ é usada} \\ 0, & \text{caso contrário.} \end{cases}$$

$$\text{Minimizar} \sum_{i=1}^{n} \sum_{j=1}^{5} c_{ij}\, x_{ij} + \sum_{j=1}^{5} F_j\, Z_j$$

sujeito a:

$$\sum_{j=1}^{4} x_{ij} = 1 \; (i = 1, \ldots, n)$$

$$\sum_{i=1}^{4} r_{ij} x_{ij} \leq b_j\, Z_j, (j = 1, \ldots, 4)$$

$$x_{ij} \in \{0, 1\} \; (i = 1, \ldots, n) \; (j = 1, \ldots, 4)$$

$$Z_j \in \{0, 1\} \; (j = 1, \ldots, 4)$$

Algumas linguagens baseadas em grafos tentam representar modelos matemáticos em um nível conceitual bem mais alto do que aqueles sobre as variáveis de decisão e restrições, chamada LPFORM. Neste formato, os vértices representam variáveis de decisão enquanto os arcos representam também variáveis de decisão (atividades de transporte, levar produto

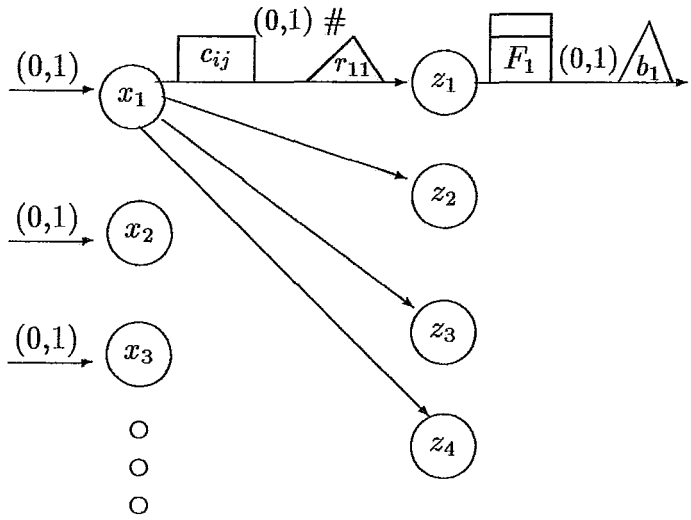


Figura II.3: Representação NETFORM de um modelo de Localização Capacitado.

da fonte ao destino) ou outras restrições lógicas que não representadas diretamente no problema linear. Os vértices também podem representar sub-modelos existentes que estão sendo usados no modelo maior. [Murphy et al 92] mostraram que a LPFORM generaliza as representações NETFORM. Na figura II.4, tem-se um grafo LPFORM para a representação do problema clássico de transporte. Cada vértice quadrado representa um conjunto de atividades, e o arco representa o fluxo de produto da fonte ao sumidouro.

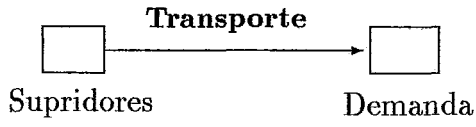


Figura II.4: Representação LPFORM do modelo de Transporte.

A modelagem estruturada e a representação baseada em grafos por modelos provêem os mecanismos formais mais robustos para a representação de modelos. Porém a modelagem estruturada representa uma tentativa ambiciosa de determinar um formalismo padrão de modelagem, [Jones 94].

3. <u>Coleta de Dados</u>

Uma formulação sem dados não pode ser resolvida. Em muitos problemas reais, os dados já existem em bases de dados corporativas, planilhas, etc. Algumas vezes, os dados e a formulação devem ser combinados de tal modo que um algoritmo possa ser aplicado. Em geral os sistemas gerenciais estão permitindo ligações com bases de dados relacionais, *Structured Query Languages*

(SQL) e os vários formatos de arquivos das planilhas. A disponibilidade e acessibilidade são elementos cruciais no uso e análise de um modelo.

4. **Execução do Algoritmo**

Uma vez que o modelo conceitual tenha sido construido, o modelo formulado, e os dados coletados, o algoritmo deve ser executado. Nesta fase, a única tarefa do usuário seria de enviar um comando de execução. Entretanto, quando o algoritmo não convergir ou quando a solução é considerada ilimitada ou inviável, em geral se deseja observar a execução interna do algoritmo. Esta fase requer uma representação apropriada. A intervenção humana durante o processamento da solução, pode estar relacionada à simples visualização dos resultados com o objetivo de ajudar a compreender a execução do algoritmo, bem como pode dar novas idéias sobre o problema.

Alguns processos de intervenção podem ser colocados como:

Otimização Interativa, [Fisher 86] considera uma ligação entre as habilidades de percepção humanas com a velocidade computacional de modo a produzir uma forma híbrida de otimização (muito usada em análises de grupamentos) no roteamento de veículos, e na localização de facilidades.

Animação de Algoritmo, À medida que o algoritmo é executado, uma figura sobre o estado atual do algoritmo (forma da estrutura de dados, por exemplo) é atualizada quando certos eventos importantes ocorrem. Estas técnicas são tipicamente usadas na simulação de eventos discretos, [Grant & Weiner 86], [Bell & O'Keefe 87]. Em tal animação, uma figura do problema muda de forma a certos eventos no tempo. Apesar de muitos autores considerarem sobre a importância da animação com o objetivo de depurar e validar um modelo, ela não retira a necessidade de análises estatísticas cuidadosas, uma vez que uma animação simples não poderia incluir exemplos de comportamento importante, [Bell & O'Keefe 87].

Muitas técnicas porém, têm obtido sucesso devido à sua representação gráfica natural. Um exemplo clássico vem da programação matemática e o problema de fluxo de custo mínimo, que com o uso de animação fica muito mais clara a percepção dos envios de fluxo entre os diversos vértices da rede.

A visualização de algoritmos pode levar a novos problemas e teorias, isto aconteceu com Hubbard que visualizava o comportamento do modelo de Newton para encontrar raízes de funções não lineares. Seu interesse era de saber como a solução resultante dependia do ponto inicial. Quando ele colocou cores relativas a cada ponto inicial baseado na solução final que era produzida, a imagem final representava um fractal, [Jones 94].

5. **Análise da Solução e Apresentação dos Resultados**

Uma vez executado o algoritmo, uma solução é obtida. Para problemas de programação linear, além da solução uma demonstração da sensibilidade da solução é apresentada, incluindo os valores duais e os limites sobre os coeficientes da função objetivo e sobre as restrições.

Em muitos casos a solução não está correta e um inumerável conjunto de problemas podem ocorrer devido a não se ter obtido uma solução satisfatória. Muita pesquisa vem sendo realizada com o objetivo de prover uma melhor análise das soluções e diagnosticar inviabilidades, [Fiacco 90], [Chinnek & Dravnieks 89].

Uma observação importante colocada por [Jones 94] considera que a depuração de um modelo, apesar de muito importante, não garante que esteja correto. Os modelos devem ser validados em relação a um comportamento julgado a partir dos padrões do mundo real. Ou seja, se o resultado não corresponde às expectativas, talvez o modelo não tenha coberto alguma importante face do problema real. Ou talvez o modelo seja simplesmente impreciso. As ferramentas de depuração podem de fato ajudar a provar o comportamento do modelo, mas fica mais vulnerável à formalização da validação. O modelo foi criado para representar um problema complexo do mundo real, em virtude disso, como se pode estar certo de que o modelo representa o mundo real com precisão suficiente?

II.2 Sistemas Baseados em Grafos

Assim como os modelos matemáticos podem ser representados por formulações algébricas, muitos problemas podem ser modelados através de grafos de atribuição. Grafos de atribuição são grafos cujos vértices são particionados em tipos, com valores ou atributos associados. Esta forma de representação é muito útil a uma diversidade de problemas, principalmente àqueles relacionados à otimização em grafos.

Sistemas de Modelagem Baseados em Grafos (SMBG) consideram em geral o tipo do grafo ou o *ESQUEMA* do grafo, colocando as características de uma dada classe de grafos. O programa que cria um esquema do grafo é chamado de Editor de Esquema, [Jones 90], [Jones 92]. Em geral, o **Editor de Esquema** está relacionado a problemas específicos, ou seja o problema de transporte e o problema de planejamento de projetos (PERT/CPM) possuem editores de esquema que estão adaptados às regras de funcionamento de cada problema específico. O objetivo é de reduzir principalmente os erros de modelagem que podem ser inseridos na formação das instâncias. Já o *Editor de Instâncias* permite que se defina para cada esquema os valores necessários ao modelo formalizado, gerando automaticamente o ambiente de trabalho necessário para manipular com grafos de qualquer tipo.

A idéia então, é que um projetista define claramente os aspectos do modelo, variáveis, restrições, limites e dados, e regras de formação do grafo de atribuição; enquanto que um modelador (usuário), entra com as informações sobre o modelo e verifica resultados para cada instância, figura II.5.

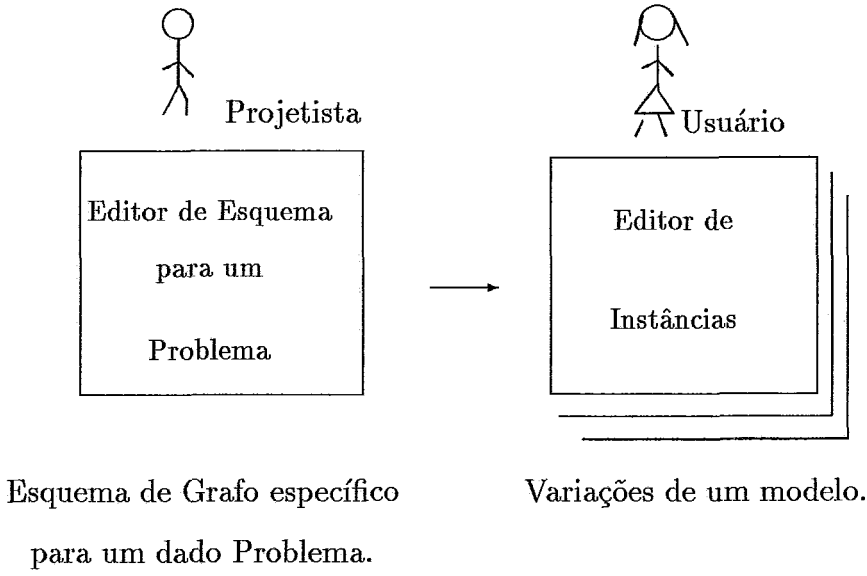


Figura II.5: Processo sistêmico em um SMBG.

As características inerentes ao **Editor de Esquema** são:

- Descrição dos diferentes tipos de vértices e ligações;
- Atributos, valores iniciais associados a diferentes tipos de vértices e ligações;
- Informação Básica, descreve como um grafo de um esquema particular (Árvore de Decisão, PERT, Roteamento de Veículos, etc.) deve ser desenhado;
- Restrições, refere-se à estrutura permitida que um grafo deve ter;
- Uma evolução do editor de esquemas seria a possibilidade de a partir dele definir procedimentos globais algoritmicos associados a problemas específicos, [Jones 92].

Já o **Editor de Instâncias** possui as seguintes características:

- Permite adicionar ou retirar vértices e ligações de modo a construir um grafo apropriado ao problema, forçando a formação do grafo conforme as restrições estruturais do problema;

- Alterar, incluir e excluir valores sobre os atributos de vértices e arestas;
- Mover vértices e ligações;
- Alterar tipos de vértices e ligações sem que sejam desrespeitadas as condições de formação do grafo esquema.

Alguns recursos são necessários para formar editores de esquema e de instâncias realmente eficientes. A Gramática de Grafos e a Programação Orientada a Objetos podem ser aliadas nesta formulação, considerando um tratamento mais universal entre instância e esquema. A seguir trata-se mais formalmente da Gramática de Grafos, [Jones 91], [Jones 92], [Jones 93], [Jones 95].

II.2.1 Gramática de Grafos

A Gramática de Grafos surgiu do estudo de linguagens formais (textos). Consiste de um conjunto de operações de edição, ou *produções* que podem ser aplicadas a um grafo. Por estas limitações se força a que restrições estruturais sobre esquemas de grafos possam ser criadas. O princípio básico é que o grafo atual é válido antes que a produção seja aplicada, e manter-se-á válido após a produção ser aplicada, [Nagl 76], [Nagl 79], [Jones 91]

As restrições estruturais podem ser de:

1. Compilação: O editor de instâncias permite ao modelador fazer quaisquer mudanças na instância corrente, e as restrições estruturais são checadas periodicamente.
2. Sintaxe Direta: O editor de instâncias limita o conjunto de modificações que o modelador pode fazer em cada ponto enquanto edita. Admitindo que as restrições estruturais são satisfeitas antes que o modelador faça as mudanças, as restrições estruturais ainda são satisfeitas após a mudança.

Dentre as características desejadas na especificação de restrições estruturais, tem-se:

1. Completa Generalidade, Ser apto a trabalhar com qualquer esquema de grafo;
2. Suporte a Edição, O editor de instâncias deve ser capaz de ser construido automaticamente;
3. Razoável Eficiência, A detecção ou geração de grafos permitidos, pelo menos para um grande número de casos onde o algoritmo polinomial é conhecido, este deve ocorrer em tempo polinomial;

4. Fácil Aprendizado, Novos projetistas devem ser capazes de aprender a especificar as restrições estruturais de forma relativamente fácil.

Dentre as vantagens relativas ao uso de Gramática de Grafos na formalização de esquemas e instâncias produzidas por um SMBG, tem-se:

1. Teoricamente as gramáticas de grafos são equivalentes à máquina de *Turing*;
2. Admitem uma representação baseada em grafos.

Ilustrando a produção de uma gramática de grafos, a figura II.6 mostra um exemplo relacionado ao Problema de Percurso de Veículos (PPV), onde um cliente a será movido para a rota entre os clientes d e e. Se um editor de instâncias permite apenas a vantagem de inserir e retirar passo-a-passo ligações de uma solução, com algumas operações individuais, o cliente pode ser reroteado. Alguns dos movimentos que poderiam ser realizados pelo modelador seriam:

1. Retirar a ligação que conecta a e c;
2. Retirar a ligação que conecta b e a;
3. Retirar a ligação que conecta d e e;
4. Adicionar ligação conectando d e a;
5. Adicionar ligação conectando e e a;
6. Adicionar ligação conectando b e c.

Se as ações realizadas de retirar e adicionar ligações são definidas por um grafo onde cada ligação corresponde a uma ação do modelador, na figura II.7 tem-se o grafo correspondente que define a produção *rerotear_cliente*.

É importante, porém, que tanto o editor de Esquemas quanto o de Instâncias tenham disponíveis um conjunto de produções a serem aplicadas convenientemente.

Na implementação de um *Editor de Instâncias* os seguintes elementos são relevantes:

1. Seleção de um esquema de grafo;

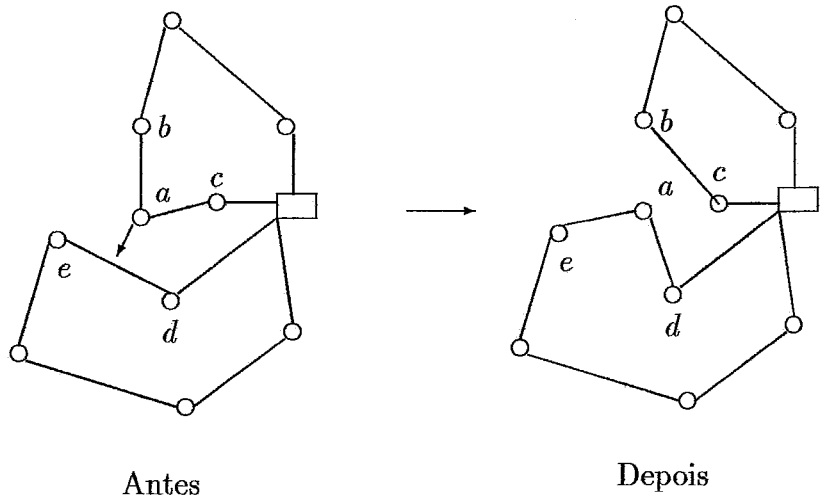


Figura II.6: Reroteando o cliente a.

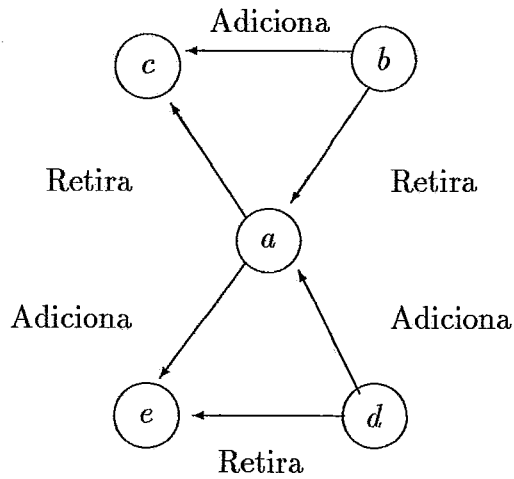


Figura II.7: Produção (simplificada) para rerotear um cliente.

2. Um menu de opções deve estar disponível apenas para o problema proposto, ou seja, haverá comandos aos quais serão divididos em dois tipos: Produções de Gramática de Grafos, e funções externas que podem ser executadas como se desejar. Em um PPV, por exemplo, as produções podem ser: Adicionar Cliente ou Depósito, Objetivo ou *Bindings*, relacionados aos comandos de edição e, Inserir Cliente em Rota, Rerotear cliente, Excluir Cliente de Rota, Adicionar Rota são referentes a operações sobre a solução.

3. O modelador pode mudar os atributos de um vértice ou de uma ligação, apontando a um vértice ou a uma ligação e em seguida selecionando um modelo de edição para mudança de atributos.

4. Quando um ambiente computacional permite múltiplas instâncias e múltiplos problemas podendo ativá-los em paralelo, ou seja, supondo que se deseja rodar um problema de roteamento e outro problema de localização, enquanto o problema de roteamento trabalha (executa) o de localização também estará trabalhando. Quando os resultados estiverem disponíveis serão exibidos diretamente nas suas respectivas janelas. Neste caso, apenas sistemas baseados em ambientes que permitem múltiplos processamentos (multi-tarefa) simultâneos são capazes de englobar esta ferramenta (Microsoft Windows NT e 95, Unix e Linux, OS/2, dentre outros).

Com o objetivo de invocar uma produção de Gramática de Grafos, o modelador usaria o *mouse* para posicionar inicialmente os nós e os elos aos quais a produção seria aplicada. Esta operação é chamada de seleção, figuras II.8, II.9, II.10 e II.11.

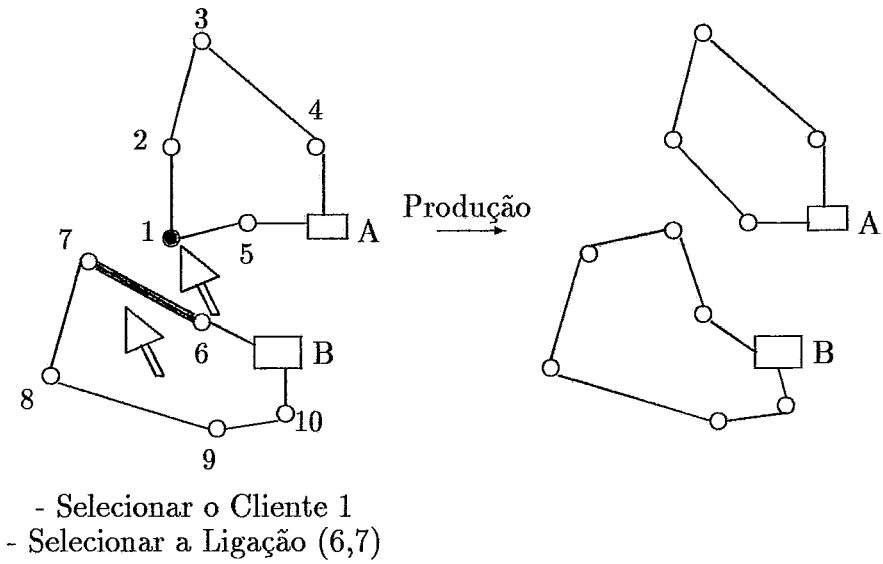
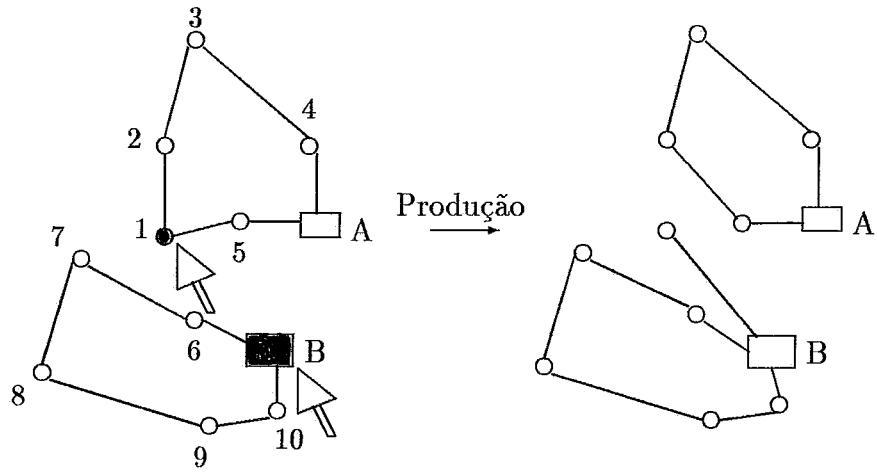
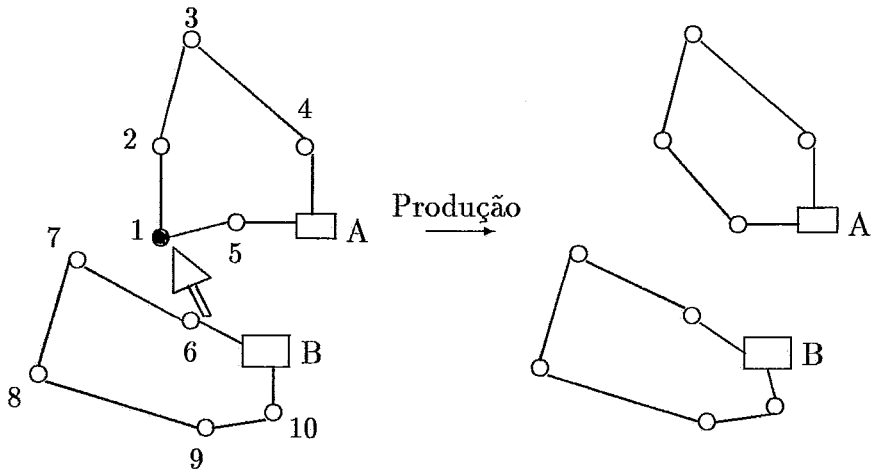


Figura II.8: Produção *Rerotear_cliente*.



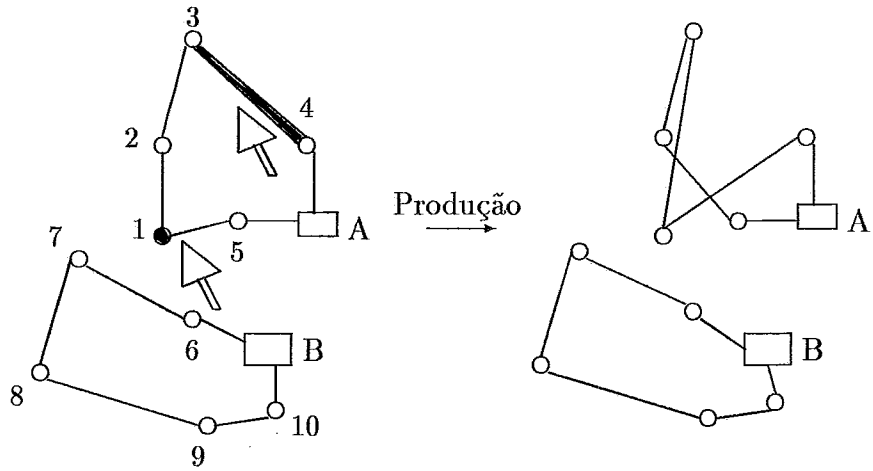
- Selecionar o Cliente 1
- Selecionar o depósito B

Figura II.9: Produção *Adicionar_rota.*



- Selecionar o Cliente 1

Figura II.10: Produção *Retirar_cliente.*



- Selecionar o Cliente 1
- Seleciona a ligação (3,4).

Figura II.11: Produção *Inserir_cliente.*

Na implementação do *Editor de Esquema*, a partir da definição de um esquema particular de grafo (Árvore de Decisão, Problema de Transporte, Roteamento de Veículos, etc.) o projetista pode definir atributos, ligações e rótulos.

1. Atributos, são informações sobre o tipo de nó ou ligação (cliente, fábrica, arco, elo), e custos relativos (deve possibilitar entrar com novos atributos);
2. Rótulos, são textos que simplesmente aparecem nos nós e nos elos;
3. *Bindings*, são as produções de Gramática de Grafos que estão associadas a um problema específico, e ao vértice ou ligação.

É importante notar que quando uma mudança é realizada em um modelo no Editor de Esquemas, todas as instâncias daquele modelo incorporarão a mudança. Para construir produções, de um menu de Esquema, o projetista selecionaria um comando de *Criar Produção* de um Esquema de Grafo, a qual constrói automaticamente um novo tipo de grafo, permite a criação de grafos representando produções de gramática de grafos. Na criação da produção *rerotear_cliente*, por exemplo, o projetista deve adicionar vértices clientes e depósitos e ligar elos quando for necessário, e mudar os valores de seus atributos.

Uma produção *Rerotear Cliente* pode ser construida como segue:

```
Producao Rerotear_Cliente
{* Descricao: Esta producao promove a transferencia de um cliente entre
duas rotas distintas em posicao definida na outra rota. }
```

```
Entrada: Cliente, Rota_Cliente
         Elo, Rota_Elo
         Vertice_i(Elo), Vertice_j(Elo)

Inicio

   { Abre Ligacoes }
   Desconectar(Pred(Cliente), Cliente, Rota_Cliente);
   Desconectar(Cliente, Suces(Cliente), Rota_Cliente);
   Desconectar(Vertice_i(Elo), Vertice_j(Elo), Rota_Elo);

   { Fecha Ligacoes }
   Conectar(Pred(Cliente, Suces(Cliente), Rota_Cliente);
   Conectar(Vertice_i(Elo), Cliente, Rota_Elo);
   Conectar(Cliente, Vertice_j(Elo), Rota_Elo)

Fim;
```

Como foi visto, grafos são estruturas atraentes na visualização de problemas complexos, uma vez que as informações são dadas através de uma visualização figurativa. Algumas técnicas importantes na visualização de grafos são:

Zoom, aumento ou diminuição da visão de um grafo;

Transladar, mover um grafo à direita ou à esquerda, acima ou abaixo;

Rotacionar, girar um grafo de modo a obter uma melhor visualização de um conjunto de vértices, em relação a um vértice definido.

Formulacão Simbólica, vários tipos de esquema de grafos apresentam uma estrutura regular a qual foi explorada durante a sua criação. Modelos podem ser colocados como blocos de outros modelos, formando daí um problema geral.

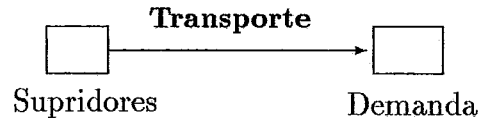


Figura II.12: Representação regular baseada no Problema de Transporte.

Na figura II.12 tem-se uma representação LPFORM regular de um problema que pode absorver instâncias de qualquer ordem. A Gramática de Grafos provê uma estrutura favorável para a representação da essência da estrutura

de um grafo. Em particular, desde que as produções de gramática de grafos descrevem como construir qualquer grafo de uma classe, elas são independentes do tamanho do grafo a ser criado.

Bindings, relacionam-se à possibilidade de visualização de algumas características de vértices e elos. Por exemplo, alguém poderia achar interessante colorir os arcos de vermelho com fluxo e de verde aqueles sem fluxo. Ou mesmo apresentar um grafo com uma outra visão, como no problema PERT/CPM, uma rede PERT ou em gráficos de GANTT (Barras). Formas (retângulos, quadrados, figuras, etc.) dadas ao vértices e ligações também são *bindings*, que podem ter um comportamento diverso e individualizado.

II.2.2 Formulação Baseada em Modelagem Interativa

[Geoffrion 87] recomendou as seguintes características para uma conceituação moderna de modelagem de sistemas:

1. Modelagem Ajustável à Comunicação Gerencial, ao uso Matemático e à Execução Direta por computador

   Uma estrutura de concepção rigorosa e coerente baseada em simples representação do formato de modelagem ajustado à comunicação gerencial, ao seu uso matemático, e à execução direta por computador. (Nos SMBGs os grafos atribuidos são a estrutura conceitual de suporte.)

2. Independência entre Modelo e Solução

   Independência entre a representação do modelo e a solução do modelo, com padrões de interface de modo a facilitar a construção de uma biblioteca de modelos e o acesso fácil a algoritmos de recuperação, sistemas de equações simultâneas, otimização e outras manipulações importantes.

3. Generalidade no tratamento da maioria dos Paradigmas de Representação de Modelos de PO

   Generalidade suficiente para englobar a maioria dos paradigmas da representação de modelos que a PO e áreas relacionadas à modelagem têm desenvolvido para a organização da complexidade da realidade (análise de atividades, árvores de decisão, fluxo em redes, cadeias de Markov, sistemas de filas, e assim por diante).

4. Utilidade na maioria dos pontos relacionados ao ciclo de vida do trabalho de modelagem

5. Representação Independente da estrutura do Modelo Geral

   Representação independente da estrutura do modelo geral e dos dados detalhados necessários à descrição específica das instâncias do modelo.

6. Implementação em Interface Estado-da-Arte

   Implementação em estação pessoal com uma interface moderna de usuário (ou seja, Visualmente Interativa, Diretamente Manipulativa, Sintaticamente Humana, e com o livre uso de gráficos e tabelas)

7. Facilidades de Banco de Dados

   Facilidades integradas para o gerenciamento de dados e perguntas específicas ao banco de arquivos, na tradição dos sistemas de bancos de dados.

8. Cálculo de expressões na tradição de planilhas e computadores pessoais

Estes passos são interessantes na construção de qualquer software do gênero e expressam a complexidade de realizá-los dentro da magnitude e abrangência das observações feitas acima. Tentativas de desenvolvimento de softwares baseados em grafos têm sido empreendidas por diversos autores, algumas delas são descritas a seguir:

[Wolfberg 69]

AGILE, [Delgrande 80]

Baseado em linguagem C, inclui novas construções e instruções em C, incluindo as construções GRAFO, Vértice, Elo, assim como conjunto. O sistema foi usado para construção de interface com o usuário, por exemplo, para animar algoritmos de fluxo em redes.

[Götler 87]-[Götler 89]

CABRI, [Dao et al 86]

Segue uma aproximação orientada a objetos onde se pode definir novos tipos de nós e elos dos já existentes, com os novos tipos herdando todas as propriedades dos tipos antigos exceto aqueles escritos por cima. Este software foi também usado para construção de parte de si mesmo, uma vez que a linguagem de programação é baseada em grafos.

NETPAD, [Dean et al 92]

Este sistema inclui uma interface de usuário modular na qual vários algoritmos de otimização combinatória podem ser inseridos.

GX, [Lee 93]

Desenvolvido em ambiente Apple Macintoch, este sistema também foi programado em PROLOG, porém com menos recursos que aquele desenvolvido por [Jones 90]. Este software incorpora idéias de preparação esquemática propostas por [Jones 92], e envia o modelo com os dados a um *solver* padrão, como o LINGO. Os resultados, nestes casos não são representados na tela, apenas em forma textual.

Sanford GRAPHBASE, [Knuth 93]

É uma biblioteca de programas em C, onde são definidas estruturas avançadas de grafos, e uma coleção de algoritmos de manipulação de grafos (conectividade, isomorfismo, ciclos, cliques, etc.) e otimização em grafos (caminhos mínimos, árvores geradoras, coloração, *matching*, etc.). Esta biblioteca é distribuida pelo autor livremente.

NETWORKS, [Jones 90]-[Jones 95];

Usa grafos de atribuição, e o processo de modelagem é feito através do editor de esquema (conforme descrito) e de instâncias. Feito em PROLOG para ambientes Apple Macintosh, possui três modos para resolver esquemas especiais que foram inseridos no sistema, um na tradição das planilhas (problemas de caminho crítico e árvores de decisão), outro incorporando habilidade na execução de perguntas a bancos de dados, especificamente orientados a grafos, e o último um algoritmo de fluxo de custo mínimo escrito em linguagem Pascal. O autor usa as facilidades do ambiente e busca através do PROLOG a tradição de Inteligência Artificial no tratamento dos grafos. Esta forma de trabalho mostra-se robusta, uma vez que a linguagem adequa-se perfeitamente à modelagem com grafos.

SisGRAFO, [Negreiros et al 92];

Usa grafos de atribuição, e o processo de modelagem é feito através do editor de instâncias. Feito em PASCAL para ambientes Microsoft DOS, possui um conjunto de algoritmos exatos e heurísticos na composição de representação dos modelos, e uma taxonomia baseada em tipos de problemas que serão avaliados por um usuário. O editor de esquema não existe, uma vez que a tarefa de preparar grafos para diferentes modelos neste software não está incluida. A idéia é mais abrangente, no sentido de que um modelo único (grafo universal) poderá ser usado por vários problemas diferentemente, cada um dentro de sua especificidade. Problemas como: árvores geradoras mínimas, caminhos mínimos, fluxo em redes de custo mínimo, atribuição, atribuição capacitada, *matching*, localização de facilidades, caixeiro viajante, carteiro chinês, carteiro rural, problema de roteamento em nós, problema de roteamento geral, dentre outros são resolvidos diretamente a partir de um dado grafo proposto pelo usuário.

NETWK, [Donohue 95];

É um software para ambientes Microsoft Windows, possui um editor de grafos o qual traduz os diagramas de redes para formulações resolvíveis de programação linear inteira mista. Trabalha com planilhas e possui um conjunto de modelos pré-estabelecidos com o objetivo de tratar os problemas em análise. Os resultados são expostos na rede de trabalho, sobrepondo de forma identificada os vértices e arestas pertencentes à solução. Uma vez que se trata de um sistema baseado em Programação Linear Inteira Mista, este software está limitado a problemas que podem ser modelados como problemas desta classe, tais como: Caminhos mínimos, árvores expandidas mínimas, atribuição, caminho crítico, fluxo multi-comodidade, o problema do caixeiro viajante, dentre outros.

II.2.3 Construção de Algoritmos em Modo de Edição

Na fase de representação de um problema, como foi exposta no item II.1.3, a necessidade de um editor de esquema baseado em gramática de grafos permite que se determinem as possíveis produções sobre o esquema de grafo a ser resolvido. Algumas produções são apresentadas em [Jones 93] para os problemas de Árvores de Decisão, onde fórmulas de atribuição em vértices são definidas com o objetivo de determinar as decisões ótimas e o máximo ou mínimo valor esperado. Isto caracteriza que em alguns casos o próprio esquema de modelagem pode ser usado com o objetivo de ajudar na formulação algorítmica da decisão. O mesmo pode ser considerado nos problemas de PERT/CPM, onde a própria estrutura conceitual do grafo já ajuda na obtenção da solução do problema através do uso de produções.

Problemas genéricos de programação linear, não linear ou programação dinâmica, as produções seriam os próprios algoritmos de resolução dos modelos através de métodos como o SIMPLEX, de Pontos Interiores, de Gradientes Conjugados, ou métodos padrões de estados de espaços. Há casos porém onde os problemas estão mal estruturados ou mesmo semi-estruturados, e aqui apenas regras heurísticas poderiam ser aplicadas.

Metaheurísticas poderiam ser definidas como elementos principais de um conjunto de algoritmos adaptativos a problemas mal estruturados ou mesmo semi-estruturados. Algumas rotinas poderiam ser chamadas com o objetivo de guiar a decisão no encaminhamento das soluções. Isto porém não é fácil de se implementar, uma vez que a composição de uma heurística baseada em Metaheurísticas passa por uma formalização que requer em geral muitos refinamentos e estratégias que dependem muito do sentimento do especialista no problema.

A definição generalizada de problemas é um passo importante, uma vez que ajuda na visualização do modelo e na correspondente obtenção dos valores

e relacionamentos da solução ótima. [Jones 93] mostra a facilidade na formulação geral de um PPL e na observação de resultados considerando as variáveis primais e duais, restrições, e objetivos primais e duais.

Uma questão importante surge: Uma vez especificado o modelo e as produções a ele referenciadas, como integrá-lo ao editor de instâncias de modo a gerar instâncias de diferentes tamanhos e valores?

A integração é pois fundamental, e é essencialmente atendida pelo uso de duas técnicas relacionadas onde:

1. Valores de Atributos podem depender de valores de outros atributos em outros modelos. Mudanças nos atributos podem se propagar automaticamente;
2. Atributos em um modelo podem se referir a outros modelos, não apenas a outros atributos. Uma vantagem suportada é que se pode construir uma interface *icônica* tal como se dá em certos tipos de sistemas computacionais gráficos.

Alguns atributos são fixos, pois esperam as dimensões da instância a ser gerada, variáveis, restrições, etc. Com eles, então, permite-se a inclusão de instâncias que variam conforme as informações dadas pelo usuário.

II.3 O Sistema SisGRAFO

O sistema SisGRAFO foi projetado inicialmente por [Negreiros et al 92], com o objetivo de ser um ambiente gráfico de solução de problemas de Otimização em Grafos. Na proposta original o sistema constitui-se de um editor de grafos (Instâncias) o qual se encarrega de gerar grafos de vários tipos (euclidianos, simétricos e mistos), que possuem atributos nos vértices e nas ligações. Os modelos gerados comunicam-se com algoritmos específicos de otimização em grafos (árvores, caminhos, fluxos, etc.), onde atributos específicos ao problema são tratados localmente pelo modelo. Um grafo solução é gerado pelo algoritmo específico onde o usuário pode interferir no resultado apenas após a sua apresentação (em módulo de alteração) para alguns dos problemas considerados.

Apesar de ter sido construido sem pretensões de ser uma ferramenta de geração de problemas, e sim de visualização de resultados a partir de um banco de problemas, a formação conceitual do SisGRAFO permite que uma gama de problemas relacionados a grafos possam ser inseridos e tratados de forma conveniente. É pois abrangente, como um Problema de Programação Linear, ou especializado em uma dada classe de problemas. Indo mais além, com poucas mudanças estruturais é

possível transformar o SisGRAFO num ambiente tão ou mais completo que aquele proposto por [Jones 90].

A abrangência conceitual do problema de formação de um grafo pura e simplesmente, passa pela construção e estruturação de uma classe grafo, que incorpora em si mesmo conceitos e procedimentos vindos da teoria dos Grafos. Basicamente, para se formar uma instância de grafo é preciso defini-lo como espécie, de modo que possa atender a diversas formulações que sejam tratáveis diretamente.

Os problemas passam por uma estruturação simbólica onde grande parte dos modelos não admitem produções pré ou pós definidas, as quais atingem rapidamente soluções confiáveis, uma vez que dependem de chamadas a métodos e relações condicionais não triviais de serem consideradas visualmente ou por simples chamadas recursivas como nos problemas de Árvores de Decisão e de Caminho Crítico.

Em virtude disso, a opção pelo conceito via *grafo universal* (uma instância de grafo que é usada por qualquer problema) e algoritmo universal (um algoritmo que possa ser adaptado a qualquer problema específico obtendo sempre soluções ótimas nos casos estruturados e próximas do ótimo nos casos semi-estruturados e não estruturados), parece uma utopia pelo menos no que concerne a problemas de combinatória. Alguma aproximação pode ser feita, e dentro do que propõe o SisGRAFO, um grafo instância poderá ser universalmente tratado por vários algoritmos específicos de forma natural, onde os atributos faltantes e características serão reclamados durante sua execução.

A seguir, discute-se inicialmente a organização e a arquitetura do SisGRAFO considerando sua estrutura principal de dados e uma proposta de restruturação de modo a viabilizar a visualização e análise em hipergrafos e grafos de larga escala. O editor de grafos é descrito considerando as ferramentas existentes relacionadas à formação do grafo e à visualização do contexto em que o grafo se insere. Os problemas tratados pelo SisGRAFO são apresentados de forma genérica, por fim se coloca as perspectivas do sistema dentro de uma visão mais generalizada, próxima a um SAD de alto nível.

II.3.1 Organização e Arquitetura

O SisGRAFO em sua versão 1.0 está projetado sobre um menu gráfico de opções tipo *pop-down*, onde três operações básicas são realizadas: Arquivos, Problemas e Configuração.

1. Arquivos, tem-se operações de criação de um novo grafo, abrir um grafo já existente, salvar, salvar como e editar um grafo e um contexto de visualização.

2. Problemas, tem-se um conjunto de problemas que estão taxonomicamente posicionados à medida que o usuário vai descendo níveis nas janelas *pop-down*. No último nível o acesso é dado à execução do algoritmo específico, figura II.13.

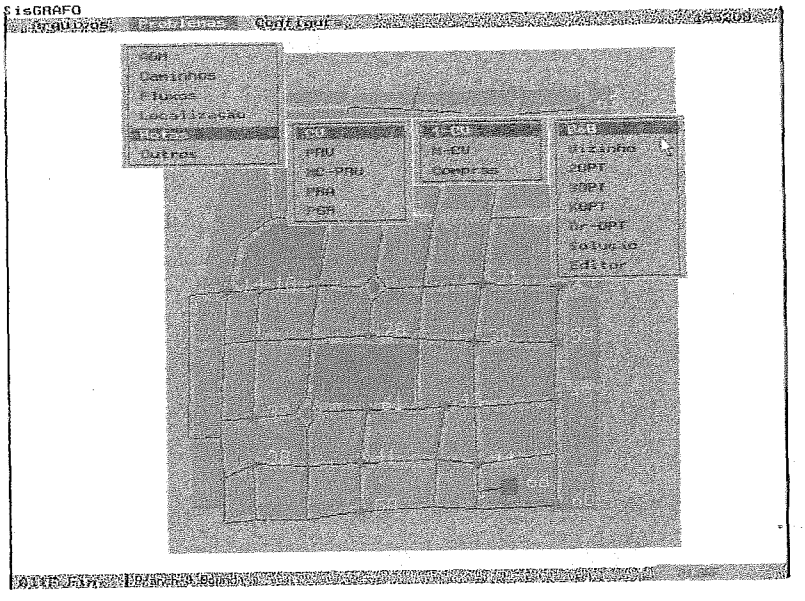


Figura II.13: Janela mostrando uma organização taxonômica de problemas no SisGRAFO.

3. Configuração, o usuário define alguns parâmetros operacionais do sistema, tais como, os locais onde estão os arquivos, os tipos de impressoras e portas de impressão, escalas do ambiente de contexto, etc.

Tipos de Grafos

Um grafo é definido no SisGRAFO como um conjunto G, formado por um conjunto de vértices ou nós, e por um conjunto de ligações entre os vértices ou nós. As ligações entre os vértices definem o *tipo* de G. Ou seja, G pode ser:

1. EUCLIDIANO: O *Grafo Euclidiano* é um grafo completo, ou seja, onde todo o par de vértices está ligado por uma aresta, onde admite-se que o custo (c_{ij}) destas arestas é o mesmo do vértice i ao j. Associado ao custo da ligação, tem-se uma métrica que o estabelece.

Sejam (x_i, y_i) e (x_j, y_j) as coordenadas (Cartesianas) dos vértices i e j de G. São métricas usuais as seguintes descritas abaixo, [Campello & Maculan 94], figura II.14.

(a) Euclidiana:

$$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

(b) Manhattan:

$$|x_i - x_j| + |y_i - y_j|$$

(c) Chebyshev:

$$\text{Max}\{|x_i - x_j|, |y_i - y_j|\}$$

(d) Afim:

$$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 - 0.5\,(x_i - x_j)\,(y_i - y_j)}$$

(e) Terrestre:

$$\alpha\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2},$$

onde α é um fator de escala.

2. SIMÉTRICO: Um *grafo Simétrico* para o SisGRAFO, é um grafo que pode ou não ser completo onde o custo $(c_{ij} = c_{ji})$ das ligações está associado a um valor real qualquer definido pelo usuário, figura II.15.

3. MISTO: Um *grafo Misto* considera a possibilidade de haver ligações direcionadas e não direcionadas. No caso de haver apenas ligações direcionadas, o grafo é puramente direcionado. Neste grafo as ligações entre os vértices i e j de G, ou são de i para j, (i, j) ou de j para i, (j, i) marcados por uma seta indicadora do sentido. Caso haja apenas ligações não direcionadas, tem-se o caso anterior, ou um grafo simétrico. Quando ocorrem os dois tipos de ligações o grafo é puramente misto, permitindo que haja uma matriz de custos entre os vértices não necessariamente simétrica, ou seja $c_{ij} \neq c_{ji}$ para alguma aresta (i, j) de G, figura II.16.

Estrutura dos Grafos

A estrutura relacionada à formação dos grafos considera o conceito de grafos atribuidos (valores) que são dados aos vértices e ligações os quais são chamados de custos. Indicadores de rótulos são dados a vértices e ligações, onde o usuário pode indicar um texto padrão a um vértice ou ligação.

Os formatos de vértices (*Bindings*) são em quatro categorias:

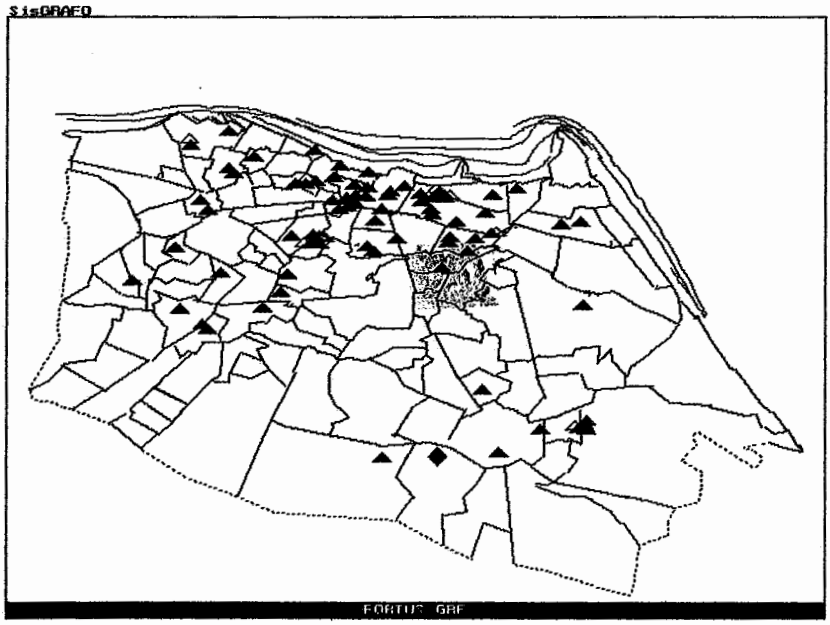


Figura II.14: Exemplo de Grafo Euclidiano.

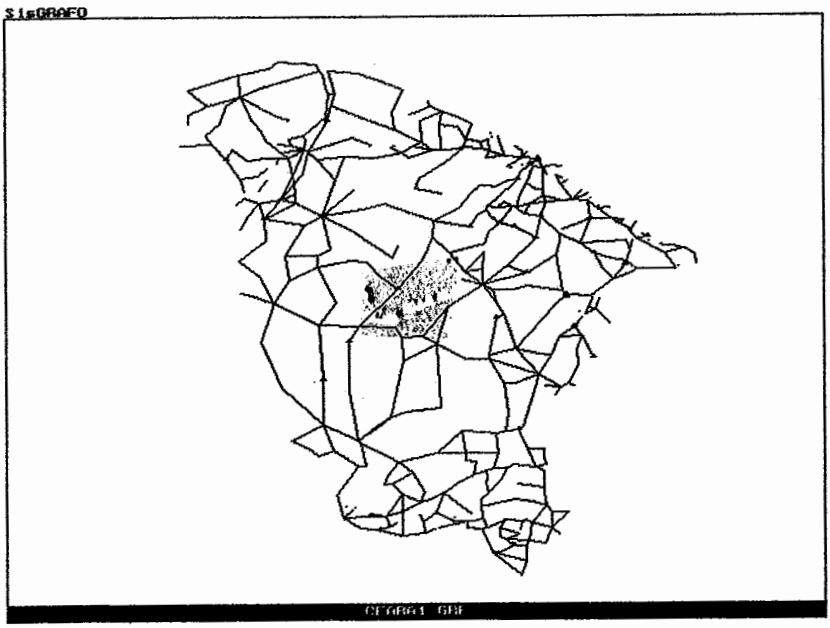


Figura II.15: Exemplo de Grafo Simétrico.

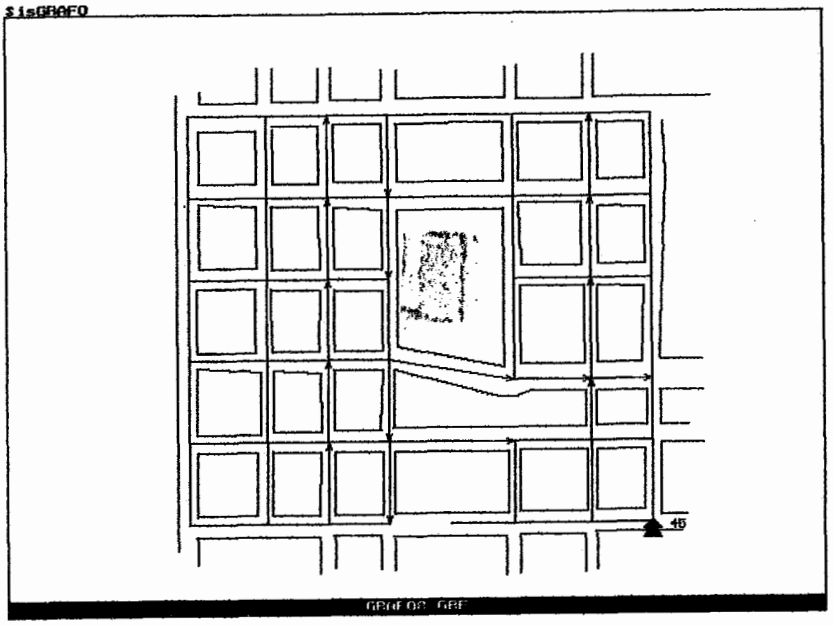


Figura II.16: Exemplo de Grafo Misto.

1. Vértices Simples, não possuem *Bindings* ou atributos de custo a ele associados, são considerados como vértices de passagem ou transbordo em grafos simétricos e mistos;

2. Vértices Origem, possuem *Bindings* sempre pintados de azul, os quais podem mudar de forma de acordo com a conveniência do usuário. Representam o ponto de partida (fábricas, depósitos, supridores, etc.) de vários algoritmos relacionados. Estes vértices podem incorporar atributos especiais, dependendo do problema a ser resolvido;

3. Vértices Cliente, são vértices sempre pintados de vermelho os quais correspondem às demandas em diversos problemas relacionados. Neste caso o conjunto de atributos pode variar enormemente de problema para problema dependendo de seu enfoque;

4. Vértices Destino, são vértices especiais que representam o sumidouro em problemas de fluxo, ou destino de um caixeiro viajante (caminho hamiltoniano), ou final constante de um conjunto de rotas especiais. Estes vértices são sempre pintados de verde e possuem atributos a ele relacionados que poderá ter um comportamento específico dependendo do problema em análise.

As ligações possuem formatos particulares dependendo do problema em análise. No caso de grafos euclidianos as ligações entre pares de vértices são subtendidas, porém nas soluções, dependendo da métrica em uso, o sistema mostra a sua representação visual (Euclidiano - reta, Manhattan - bordas de retângulos). Em grafos simétricos, apenas as ligações não direcionadas são apresentadas, onde *loops* ou réplicas de ligações não são permitidas, e uma reta ligando os vértices aparece como a representação física da ligação. Em grafos mistos, admite-se ligações não direcionadas e direcionadas, aquelas direcionadas são representadas por uma seta indicando o sentido da ligação entre dois vértices.

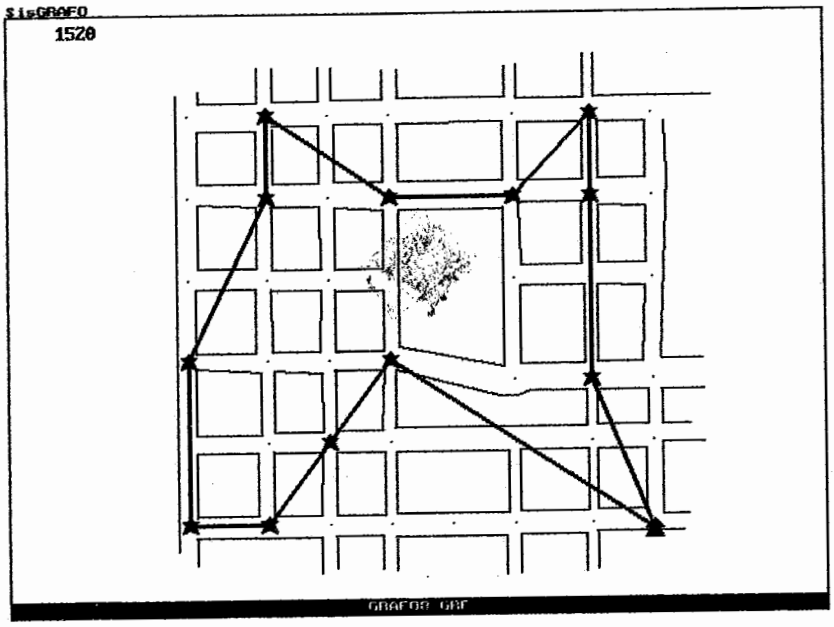
Um grafo somente pode assumir um único tipo de representação, porém é possível mudar de tipo a partir de um único. Esta condição permite que se visualize a contribuição do fator custo na formação de soluções em função das restrições de movimentação. Na figura II.17 tem-se uma visualização do efeito da presença do grafo misto no sequenciamento da rota do caixeiro viajante numa de suas instâncias. Nota-se neste caso que o sequenciamento se manteve constante nos casos euclidiano e misto. A alusão a este problema foi feita em [Potvin et al 94], no detalhamento do sistema ALTO, um ambiente para a análise de heurísticas de percurso de veículos.

Estrutura de Dados

A estrutura de dados adotada no SisGRAFO para a formação dos grafos é constituida de um vetor que aponta para listas encadeadas (ligações) e para arquivos específicos (vértices). A escolha desta estrutura se deve à flexibilidade na formação de grafos de atribuição e na edição em tempo real de vértices e ligações, figuras II.18, II.19, II.20, [Syslo et al 83].

A desvantagem desta estrutura reside no tratamento de multigrafos e hipergrafos, uma vez que a implementação está orientada a um único grafo universal, que não pode ser mesclado a outros tipos.

Uma estrutura de atribuição mais abrangente foi proposta por [Knuth 93], porém ressente-se sobre o volume de memória necessário para implementá-la. A estrutura de Knuth, usa fortemente as vantagens da linguagem C no tratamento dos conjuntos de vértices, ligações e grafo. A idéia da formação da estrutura é que os programas da biblioteca GRAPHBASE buscam uma representação do grafo que seja feita direta e totalmente na memória do computador. Os vértices aparecem em um vetor sequencial de registros chamados **VERTICE**, assim como na estrutura do SisGRAFO; e os arcos emanam de cada vértice em uma lista encadeada de registros **ARCO**. A diferença aparece então quando se inclui um registro **GRAFO**, para dar informação sobre um grafo como um todo.



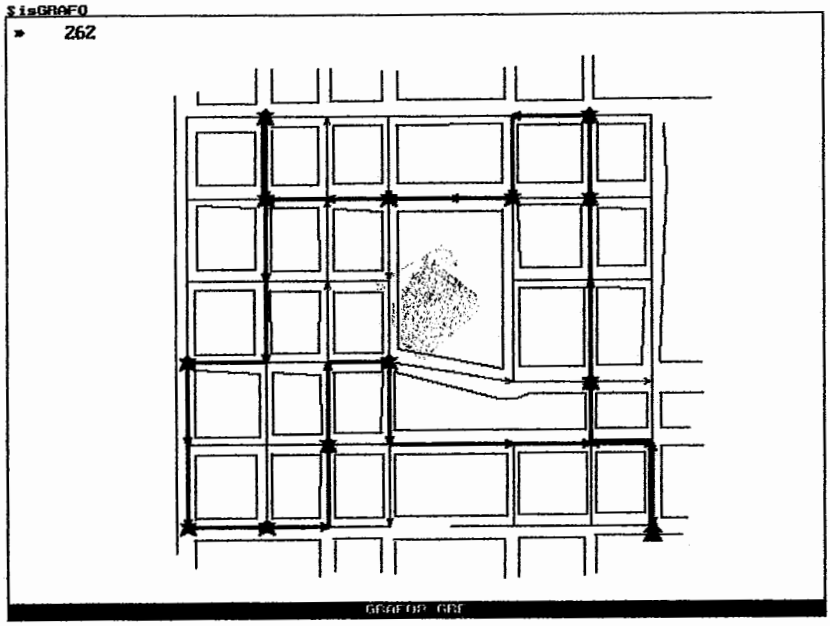


Figura II.17: Variação numa solução de caixeiro viajante euclidiano e misto, para um mesmo grafo base.

```
Type
     cc : array[1..4] of real;
     figura = (ponto, reta, quadrado, estrela, ... , outras);

     { Estrutura de Bindings }
     Bindings = Record
       cor    : word; { cor do elemento }
       tam    : real; { multiplicador de tamanho do elemento }
       forma  : figura; { indicador da figura que assume o elemento }
     end;

     { Estrutura de Vertice }
     Noh = Record
       na   : word; { Numero de arcos ligados ao vertice }
       tp   : word; { Atributo de tipo do vertice }
       rtl  : word; { Codigo relativo a um nome no arquivo de rotulos }
       rtB  : word; { Codigo relativo a um Binding no arquivo de bindings }
     cx, cy : real; { Atributo de posicao - coordenada }
      cost  : cc;   { Atributos de custo - peso }
      v, u,
      s, w  : real; { Campos auxiliares de algoritmos }
     end;

     { Estrutura de Arcos }
     Ac   : ^ Arco;
     Arco = Record
      Jarc : word; { Vertice destino }
      Tpa  : boolean; { Tipo de arco - direcionado ou nao }
      rtl  : word; { Codigo relativo a um nome no arquivo de rotulos }
      rtB  : word; { Codigo relativo a um Binding no arquivo de bindings }
      cost : cc;   { Atributos de custo - distancia, tempo, etc. }
      prox : Ac;   { Ponteiro para o proximo arco }
     end;
```

Figura II.18: Código Pascal da estrutura do SisGRAFO.

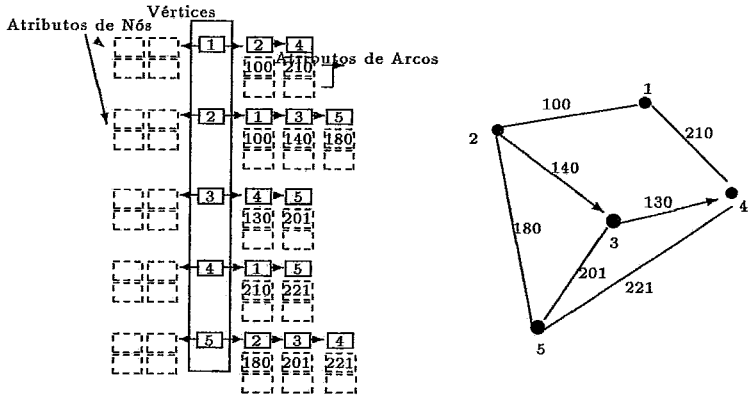


Figura II.19: Exemplo da formação da estrutura do SisGRAFO.

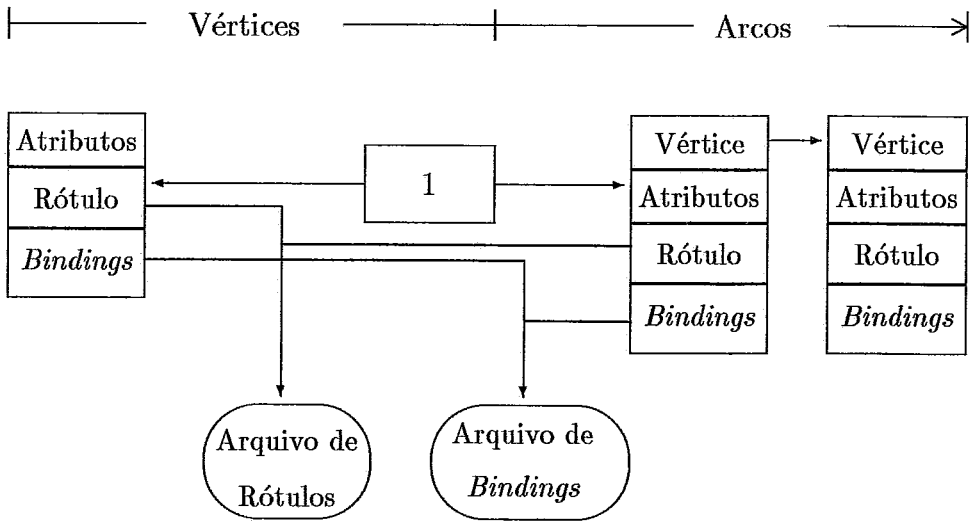


Figura II.20: Representação mais detalhada da estrutura de arquivos do SisGRAFO.

Nos *layouts* das estruturas dos registros **VERTICE**, **ARCO** e **GRAFO** inclui-se um número de campos utilitários que podem ser usados para qualquer propósito por algoritmos que manipulam os grafos. Cada campo utilitário é um tipo *union*, [Weiss 92], que pode ser tanto um ponteiro de várias espécies (vértice, arco ou grafo) quanto um inteiro ou um *string*.

A definição formal destas estruturas de dados inicia pela declaração formal do tipo *union* chamado de **util**. Os sufixos $.V$, $.A$, $.G$ e $.S$ na mesma variável, apontam para um *vertex*, *arc*, *graph*, ou *string*, respectivamente. O sufixo $.I$ indica que a variável é um inteiro.

Typedef *union* {

Struct *vertex_struct**V; /* Ponteiro para Vértice */
Struct *arc_struct**A; /* Ponteiro para Arco */
Struct *graph_struct**G; /* Ponteiro para Grafo */
Struct *S; /* Ponteiro para *String* */
Struct *I; /* Inteiro */

} **util**;

Cada vértice tem dois campos padrões para seis campos utilitários; portanto ocupando 32 bytes na memória dos sistemas, sem contar a memória necessária para o *String*. Os campos padrões são:

arcs, um ponteiro para **ARCO**;
name, um ponteiro para um *string* de caracteres.

Se um vértice v aponta para um *vertex* e $v \rightarrow arcs$ é vazio (*null*), não há arco saindo de v, e aquele registro tem um campo *next* que aponta na mesma forma as representações para todos os arcos de v.

Os campos utilitários são chamados u, v, w, x, y, z. Macros podem ser usadas para dar a eles uma forma sintática em aplicações particulares, como em células de planilhas. Estes campos são tipicamente usados para registrar por exemplo o grau do vértice ou se o vértice está marcado. Os campos utilitários também são usados para ligar vértices a outros vértices ou arcos em uma ou mais listas.

Typedef Struct *vertex_struct* {

Struct *arc_struct**A; /* Ponteiro para Arco */
Char **name*; /* *String* identificador simbólico deste vértice */
util u, v, w, x, y, z; /* Campos de múltiplos usos */
Struct *S; /* Ponteiro para *String* */

} **VERTICE**;

Cada arco tem três campos padrões e dois campos utilitários. Portanto ocupam 20 bytes na maioria dos sistemas. Os campos padrões são:

tip, um ponteiro para um **VERTICE**;
next, um ponteiro para um **ARCO**;
len, um inteiro.

Se a aponta para um **ARCO** na lista de arcos do vértice v, ele representa um arco de comprimento $a \rightarrow len$ para v e $a \rightarrow tip$ é o vértice do próximo arco de v na lista $a \rightarrow next$.

Os campos utilitários são chamados a e b, e a estrutura é a que segue:

Typedef Struct *arc_struct* {

Struct *vertex_struct*tip*; /* Arco aponta para este vértice */
Struct *arc_struct*next*; /* Outro arco aponta ao mesmo vértice */
Long **len*; /* *Comprimento* deste arco */
util a, b; /* Campos para múltiplos usos */

} **ARCO**;

No processamento dos grafos, toda a alocação de memória é feita dentro da perspectiva das listas.

Um tipo **GRAFO** é uma estrutura de dados que pode ser passada a um algoritmo que opera sobre grafos, para encontrar a árvore expandida mínima, componentes conexas, ou o que for.

Um registro **GRAFO** na GRAPHBASE tem sete campos padrões e seis campos utilitários. Os campos padrões são:

vertex, um ponteiro para um vetor de registros **VERTICE**;

n, o número total de vértices;

m, o número total de arcos;

id, um identificador simbólico que define parâmetros de procedimento GRAPHBASE que foram gerados para este grafo;

util_types, uma representação simbólica dos tipos de dados em campos utilitários.

data, uma área usada para o armazenamento de arcos e *strings*

aux_data, uma área usada para informação auxiliar que alguns usuários podem querer descartar.

Typedef Struct *graph_struct* {

Struct *vertex**VERTICE; /* Iniciando no vetor de vértices */
Long **n*; /* *Número* total de vértices */
Long **m*; /* *Número* total de arcos */
Char **id*; /* Identificador GRAPHBASE */
Char **util_types*; /* Uso de campos utilitários */
Area *data*; /* Blocos de dados principais */
Area *aux_data*; /* Blocos de dados principais */
util uu, vv, ww, xx, yy, zz; /* Campos de múltiplos usos */

} **GRAFO**;

Um detalhe importante sobre esta estrutura, é que toda a busca elementos de um grafo, ou um grafo, pode ser pesquisado em modo *hashing*, permitindo assim uma grande velocidade na busca das informações específicas sobre a memória. Outros detalhes, tais como: coleta de lixo, alocação de memória dinâmica, são um elemento forte no conceito operacional da GRAPHBASE.

A estrutura proposta por Knuth é de extrema robustez e abrangência, no que concerne a operacões relacionadas diretamente na memória principal. Porém, considerando certos ambientes de processamento, e o uso geral de memória solicitado por problemas de otimização combinatória, acredito não ser necessário manter sempre estruturas tão abrangentes. A idéia de uma estrutura adaptada ao problema me parece mais natural.

A idéia de múltiplos grafos, porém, é um avanço considerável. Isto pode ser feito em tempo de execução, uma vez que toda a memória principal dos programas pode ser tratada em blocos haja vista as limitações dos tamanhos de vetores que podem ser abertos em memória principal, para algumas linguagens de microcomputadores. O tratamento do problema de alocação de memória para arcos e coleta de lixo de memória também pode ser extendido de forma natural, porém realizado de forma complementar.

Uma nova estrutura pode ser então criada da estrutura principal inicialmente proposta pela versão 1.0 do SisGRAFO. Um vetor de registros *_Graph* pode ser considerado da mesma forma que os registros propostos por Knuth. Formulando em linguagem Pascal, tem-se:

```
Const NG = 100; { Numero de grafos - camadas de grafos }

Type

   _Graph = Record
      g        : word; { Numero de Vertices }
     id        : word; { Identificador para tipo do grafo }
   vertices : ^Vert; { Estrutura primaria }
   end;

   Grf : ^ Array[1..NG] of _Graph;
```

A estrutura de arcos seria então modificada apenas com a inclusão de um identificador de grafo, mantendo os demais elementos constantes. Com isto, todo o poder dado a esta nova estrutura tem o mesmo fundamento dado pela formulação proposta por Knuth colocada acima.

A dificuldade, porém, está no tratamento algorítmico de porções mescladas do grafo, cuja solução demanda um certo esforço individual na construção. Ferramentas e produções (gramática de grafos) de controle destas camadas para interrelacionamentos entre elas devem ser criteriosamente desenvolvidas com o objetivo de tratar os grafos ativos (camadas ativas) no encaminhamento da solução de problemas específicos.

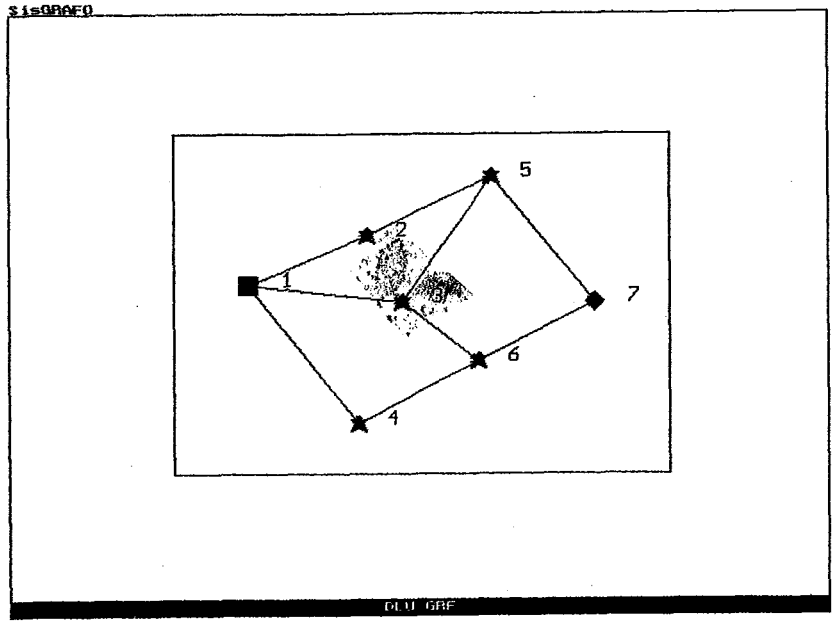
II.3.2 Editor de Grafos e Textos

Com a estrutura acima, é possível definir um gerador de grafos que possa ser acessível a qualquer momento por qualquer problema. O *editor de grafos* é em princípio gráfico, ou seja, permite ao usuário observar o posicionamento dos seus elementos a qualquer instante e gerar seus modelos conforme sua necessidade.

A vantagem básica do editor de grafos é permitir a criação, gravação, recuperação e alteração de um grafo em tempo hábil. Também, é preciso que ele seja armazenado de um modo compreensível e transportável por qualquer usuário em um formato padrão de leitura.

No SisGRAFO, os dados do grafo são armazenados (salvos) em um arquivo texto auto-explicativo, onde se tem grande parte da estrutura de dados proposta. Os rótulos e formatos são intrínsecos ao sistema, os quais estão armazenados em arquivos específicos de informação e acessíveis a qualquer instante, figura II.21.

Facilidades como deslocamento lateral, rolagem de tela, efeitos de




```
7 2  --> No. de Nohs, Tipo do Grafo
3 147 193  1  0  0.000  0.000   0   0 --> Noh  1: No. Arc, XO, YO, Tp, RTL, Custos
2 1   11.00000   0.00000     42.00000      0.00000
3 1    5.00000   0.00000     11.00000      0.00000
4 1   16.00000   0.00000     33.00000      0.00000
1 240 154  3  0  8.000  0.000   0   0 --> Noh  2: No. Arc, XO, YO, Tp, RTL, Custos
5 1   10.00000   0.00000     26.00000      0.00000
2 268 206  3  0  6.000  0.000   0   0 --> Noh  3: No. Arc, XO, YO, Tp, RTL, Custos
5 1    3.00000   0.00000     11.00000      0.00000
6 1    4.00000   0.00000     11.00000      0.00000
1 233 301  3  0  10.000 0.000   0   0 --> Noh  4: No. Arc, XO, YO, Tp, RTL, Custos
6 1    5.00000   0.00000     33.00000      0.00000
1 337 108  3  0  4.000  0.000   0   0 --> Noh  5: No. Arc, XO, YO, Tp, RTL, Custos
7 1    7.00000   0.00000     50.00000      0.00000
1 327 252  3  0  4.000  0.000   0   0 --> Noh  6: No. Arc, XO, YO, Tp, RTL, Custos
7 1    9.00000   0.00000     55.00000      0.00000
0 417 206  2  0  0.000  0.000   0   0 --> Noh  7: No. Arc, XO, YO, Tp, RTL, Custos
```

Figura II.21: Representação de armazenamento em arquivo texto de um grafo DLU.GRF do tipo misto produzido pelo SisGRAFO.

dimensões em até dez camadas, foram inseridas no contexto do editor de grafos. Estes recursos geralmente oneram o sistema em termos de ocupação de memória e de capacidade de solução dos algoritmos em ambientes limitados de processamento. Isto foi sanado através de métodos de gerenciamento de memória e arquivos em microcomputadores. A figura II.22 mostra o ambiente de edição de desenhos e grafos.

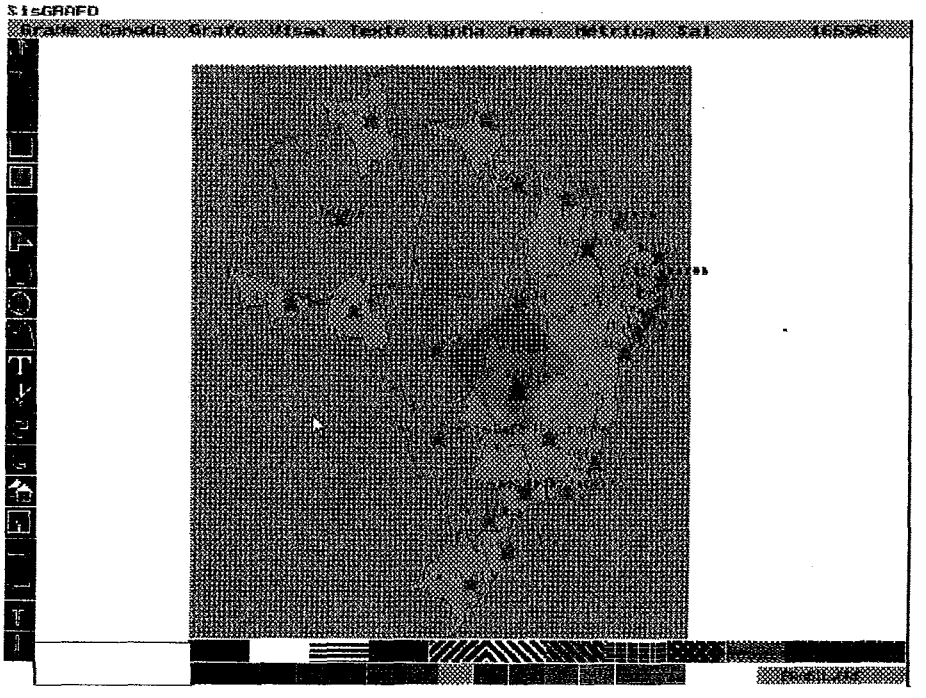


Figura II.22: Ambiente de edição de figuras e grafos do SisGRAFO.

Também é indispensável a presença do editor de textos para visualização das soluções obtidas. Este editor, proveniente das bibliotecas *Turbo Vision* do Turbo Pascal 7.0, foi ajustado às necessidades operacionais do sistema, com o objetivo de agilizar a compreensão das soluções visuais encontradas. Com ele, é possivel avaliar novas instâncias de um problema sem que para isto seja necessário sair do sistema.

II.3.3 Visualização das Soluções

Interação Editor e Algoritmos

Como foi colocado, a grande vantagem de um SMBG, é a possibilidade de interagir com problemas distintos através do uso de uma estrutura flexível e

II.3.3 Visualização das Soluções

Interação Editor e Algoritmos

Como foi colocado, a grande vantagem de um SMBG, é a possibilidade de interagir com problemas distintos através do uso de uma estrutura flexível e abrangente. Sua tarefa deve ser de agilizar a manipulação das informações a fim de permitir uma análise de sensibilidade de soluções através da manipulação física do grafo de trabalho, no qual se fará todo o processo de ajuste das variáveis do modelo geral.

O editor gráfico (de instâncias), deve estar disponível a todo momento, seja qual for o algoritmo utilizado. Ao mesmo tempo os algoritmos visualizam as mudanças ocorridas no grafo realizadas no editor, fazendo o possível para que sejam reduzidas o número de iterações de reconhecimento da nova instância de trabalho (modelo).

No SisGRAFO isso é possível em função da modularidade do sistema pelo uso de *TPUs* e *Overlays* do Turbo Pascal[2] para o ambiente Microsoft DOS. Esta vantagem, favorece o posicionamento do conjunto de bibliotecas do editor gráfico, em qualquer algoritmo usado pelo sistema, com retorno automático. Em ambiente Microsoft Windows, uma reestruturação do SisGRAFO está prevista, mantendo os princípios regulares do sistema em DOS, porém explorando a facilidade das OWLs (Biblioteca de Objetos Windows) para manipulação de interface gráfica, das DLLs (Bibliotecas de Conexão Dinâmica) cujas rotinas e funções podem estar escritas em qualquer linguagem de programação as quais são acessíveis pelo sistema, das DDEs (Troca de dados dinâmica) e DDIs (Interface de Múltiplos Documentos) com o objetivo de considerar a possibilidade de realização de múltiplas tarefas, [Swan 93].

Saída dos Resultados

Uma forma de visualizar graficamente um resultado, é obter dele uma representação convincente da solução. A representação gráfica de uma solução para um dado problema pode ser ambígua, principalmente quando há várias soluções a serem visualizadas sobre um mesmo grafo. Uma maneira de acabar com as redundâncias ou ambiguidades visuais, seria mostrar o resultado obtido graficamente, em conjunto aos impressos ou segmentados na tela gráfica, figura II.23.

Imaginando, por exemplo, o problema de k-caminhos distintos entre pares de vértices de um grafo. É possível que um mesmo conjunto de ligações sejam utilizadas por caminhos distintos, e seria difícil visualizar um único caminho ou identificar de quê caminho faria parte um conjunto de ligações.

[2] O Turbo Pascal é um compilador desenvolvido pela Borland International

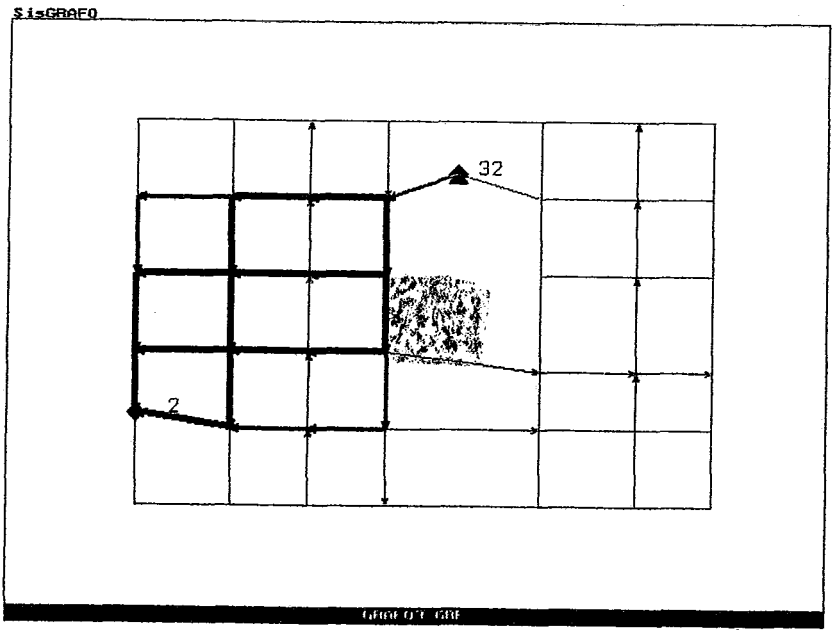


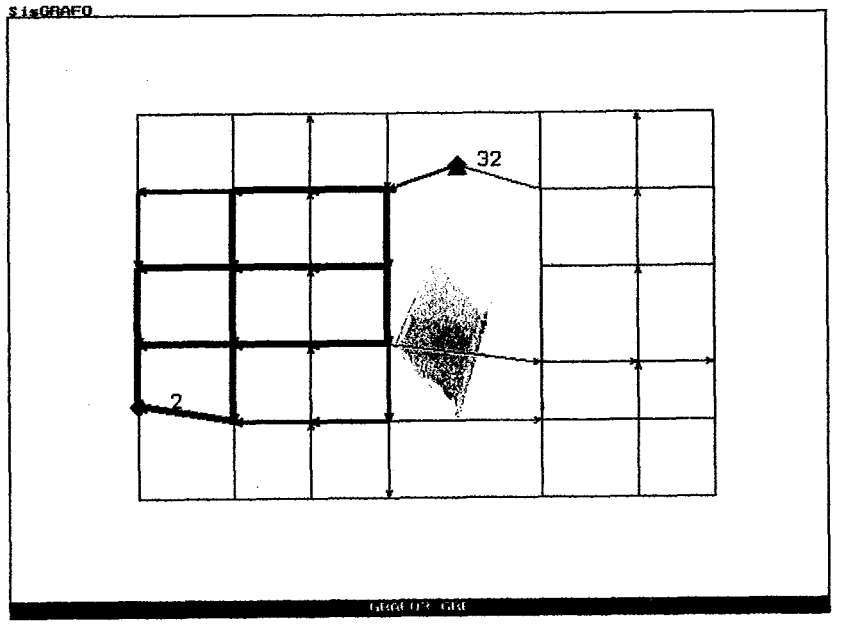
Figura II.23: Exemplo de uma saída ambígua para um problema caminhos iguais.

Assim a melhor maneira de não permitir esta redundância, é fazer com que o grafo seja apresentado junto à sua solução em cada caso particular, ou seja, uma a uma com detalhes sobre os valores relevantes obtidos da solução. Uma visualização global deve ser obtida também, e caso não seja possível observar as soluções uma a uma, um relatório simplificado deve acompanhar a solução, a fim de permitir a compreensão.

No SisGRAFO, é possível obter as soluções dos problemas gerados a qualquer instante, a partir dos resultados computados. No caso, o sistema fornece um relatório que consta de uma saída gráfica representando a solução do modelo em uso, e um *relatório* apresentando a solução numérica obtida, com dados gerais do problema, figuras II.24 e II.25.

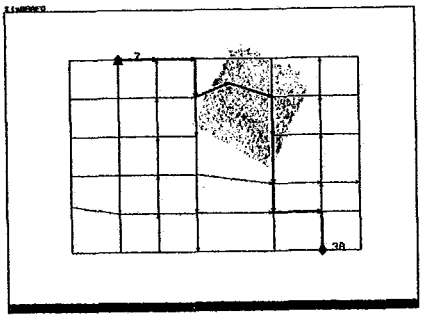
Recursos de Animação são uma outra forma de visualização mais sofisticada que foram inseridas no SisGRAFO. Uma vez que é crucial a ajuda na compreensão das soluções em problemas como caminhos, rotas e fluxos, foi inserido no SisGRAFO um elemento que permite avaliar os percursos tomados pelas soluções propostas através de um elemento que se move na tela rastreando o percurso realizado, ou as quantidades de fluxo relativo que passa por cada arco da rede.

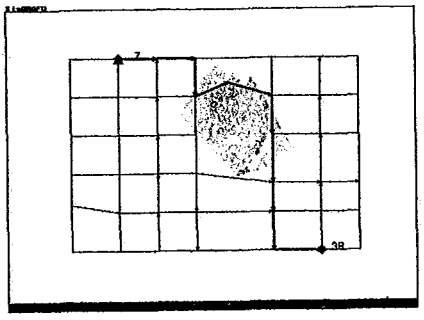
Editor de Soluções




```
SisGRAFO                                                        Versao 1.0
>>>>>>>>          CAMINHOS MINIMOS EMPATADOS                    <<<<<<<<<<

Grafo  : GRAFO3.GRF
#Nos   :   44
Tipo   : Misto

>>>>>>>>                      SOLUCAO                           <<<<<<<<<<

<-(   2  11  10   9   8  17  20  32)<-
<-(   2  11  10   9  16  21  20  32)<-
<-(   2  11  10  15  22  21  20  32)<-
<-(   2  11  14  23  22  21  20  32)<-
<-(   2   3  10   9   8  17  20  32)<-
<-(   2   3  10   9  16  21  20  32)<-
<-(   2   3  10  15  22  21  20  32)<-
<-(   2   3   4   9   8  17  20  32)<-
<-(   2   3   4   9  16  21  20  32)<-
<-(   2   3   4   5   8  17  20  32)<-

Custo Total do(s) Caminho(s) de   32 a    2:   62.000


>>>>>>>>>>>>>>>> Tempo de Processamento:  0: 0: 1.15 <<<<<<<<<<<<<<<<<<<<<<
```

Figura II.24: Impressão gráfica do resultado e *relatório* explicativo da solução numa situação de ambiguidade.






```
SisGRAFO                                                    Versao 1.0
>>>>>>>>              K-CAMINHOS MINIMOS                    <<<<<<<<<<

Grafo  : GRAF03.GRF
#Nos   :   44
Tipo   : Misto

>>>>>>>>                     SOLUCAO                        <<<<<<<<<<

<-(  38  37  27  28  29  30  32  20  19  18   7)<-
0  1o.-Caminho MAIS CURTO de     7 a   38 tem     73.000 u.c.

<-(  38  26  27  28  29  30  32  20  19  18   7)<-
0  2o.-Caminho MAIS CURTO de     7 a   38 tem     81.000 u.c.


>>>>>>>>>>>>>>>> Tempo de Processamento:  0: 0: 3. 2 <<<<<<<<<<<<<<<<<<<<<
```

Figura II.25: Impressão gráfica do resultado e *relatório* explicativo das soluções obtidas.

Com o intuito de aumentar a credibilidade do usuário nos resultados obtidos, permitindo que ele gere suas próprias soluções, ou até mesmo que ele tenha acesso a uma mudança no formato dos resultados, alguns problemas possuem um módulo de usuário, onde ele pode interagir com o problema do modo que lhe convier. Desta forma é possível ceder lugar ao tomador da decisão, para que ele mesmo verifique o quão sensível é a solução obtida em função das mudanças de atributos na instância de grafo em uso.

Um editor de soluções mais robusto que se adeque a cada problema específico está sendo projetado. Em problemas tais como roteamento de veículos e agrupamento capacitado, a importância deste editor é considerável, uma vez que ele poderá permitir ao usuário alterar os resultados produzidos pelos algoritmos de forma mais confortável e de acordo com as suas necessidades de decisão. As produções que serão incluidas neste contexto são baseadas em algumas daquela definidas por [Jones 90].

II.3.4 Ambiente Computacional, Programação e Limitações

Haja vista a idéia de disseminar o estudo de Otimização Combinatória em Grafos no meio acadêmico, como também pelo largo uso de computadores pessoais com capacidades gráficas evoluidas, optou-se pelo desenvolvimendo do sistema **SisGRAFO** inicialmente em ambiente Microsoft DOS, na linguagem Turbo Pascal versão 7.0, Borland International.

A facilidade no manuseio desta linguagem permitiu a opção pela geração do código do SisGRAFO nesse ambiente, apesar de que uma migração para o ambiente Microsoft Windows e em seguida para qualquer sistema operacional (UNIX, OS/2, Apple Macintoch), estar sendo planejada.

Devido às limitações operacionais do ambiente Microsoft DOS e do Turbo Pascal (DOS), e relativamente ao usuário final que se deseja atingir, o código foi escrito com a capacidade de gerenciar uma rede com até 500 vértices para as aplicações acadêmicas. Uma vantagem será inserida no sistema, cujas dimensões máximas dependerão exclusivamente da quantidade de memória livre disponível no computador do usuário.

Para grafos simétricos e mistos, a possibilidade de resolução de problemas com acentuado número de arcos dependerá pois exclusivamente do tamanho da memória disponível. Calcula-se que é possível gerar em memória RAM grafos com no máximo 7.500 arestas na versão acadêmica em DOS.

Um fato relevante é que o gerenciamento de problemas se torna cada vez mais proibitivo à medida que há um crescimento acentuado das redes (vértice e

ligações). Apesar dos algoritmos implementados serem quase todos de complexidade polinomial, $O(n^k)$, onde n é o número de vértices da rede e k (≤ 4), os resultados obtidos em virtude da rapidez no processamento e na apresentação dos resultados e manipulação do grafo pela interface gráfica permite apenas um meio termo entre as vantagens e desvantagens na obtenção de soluções de alta qualidade. Um grande volume de dados certamente não tem sentido de ser processado em ambientes de microcomputadores pessoais, porém a questão tamanho do volume de dados tratável depende muito do problema e do algoritmo usado.

II.4 Principais Problemas Enfocados

Dentre os problemas tratados pelo SisGRAFO, tem-se a seguinte classificação abaixo:

1. Árvore Geradora Mínima (AGM)

   O problema AGM se refere à possibilidade de construir uma árvore geradora de custo mínimo em um dado grafo simétrico. As AGMs, podem ser aplicadas em estudos de redes de telecomunicações, redes de saneamento, análise de conglomerados, dentre muitos outros, ver exemplo de AGM na figura II.26.

   O SisGRAFO resolve problemas de AGM considerando grafos do tipo Euclidiano, Simétrico e Misto (árvores enraizadas). Utiliza-se do algoritmo PRIM, [Syslo et al 83], o qual define uma árvore mínima para um grafo completo. Uma adaptação foi feita neste caso para grafos do tipo simétrico, onde uma solução bastante melhorada é atingida considerando uma aproximação a um problema de Steiner, [Christofides 78].

2. Caminhos

   O problema de caminhos se refere à possibilidade de se atravessar com custo mínimo, um conjunto de ligações de um dado grafo (direcionado ou não), a partir de um vértice origem a um destino previamente escolhidos.

   Supondo que são dados como atributos de uma aresta dois custos, onde o primeiro se refere ao custo em distância, e o segundo ao custo em tempo, o caminho mais curto será relativo ao custo das distâncias e o caminho mais rápido será relativo ao custo dos tempos. Com esta primeira divisão no uso de caminhos entre pares de vértices, pode-se obter no SisGRAFO as seguintes soluções:

   (a) 1-Caminho;

   (b) Iguais, "Todos" (se houver memória para isto) os caminhos empatados entre pares definidos de vértices de um grafo;

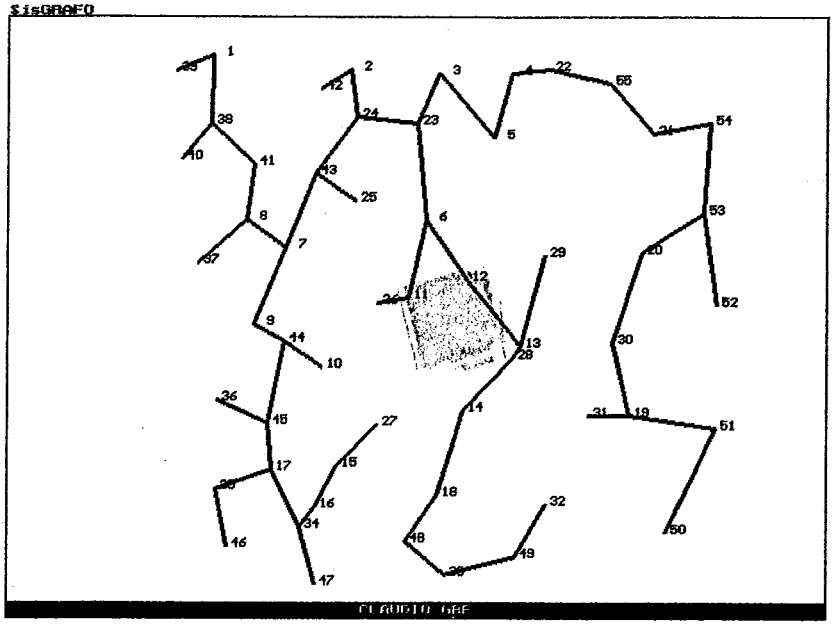


Figura II.26: Exemplo de aplicação de Árvore Geradora Mínima.

(c) k-Caminhos, Determina os k (definido) menores caminhos distintos entre pares de vértices de um grafo sem ciclos de primeira ordem;

(d) Caminhos Restritos, Considera que há movimentação proibitiva entre pares de arestas (assim como nas restrições de tráfego).

Os algoritmos de caminho foram especialmente adaptados para atender às exigências acima. A base do algoritmos utilizados são o algoritmo de [Dijkstra 59] e de Pape, d' Esopo & Moore [Syslo et al 83], que permitiram adaptações a todas as variações de caminho consideradas em tempo razoável. Uma adaptação ao algoritmo proposto por [Easa 85] para percursos com restrições de movimentação foi também realizada.

3. <u>Fluxos</u>

Estes são os problemas mais clássicos da Otimização Combinatória. Dado que se deseja enviar uma quantidade de uma determinada matéria de um vértice da rede (Fonte) a outro (Sumidouro), e dado que cada aresta do grafo possui uma capacidade mínima e máxima de fluxo de matéria que pode por ela fluir, além de um custo a ela associado, o objetivo é obter as seguintes respostas:

(a) *Fluxo Máximo*: Qual é a máxima quantidade de matéria que pode ser enviada da fonte ao sumidouro? (e como?)

(b) *Fluxo de Custo Mínimo*: Para uma dada quantidade de matéria enviada da fonte ao sumidouro, qual o menor custo total de envio? (e como?)

Os problemas de Transporte e Transporte com Transbordo, podem advir de uma transformação do problema de Fluxo de Custo Mínimo, [Jansen & Barnes 90]. Tal transformação é auxiliada pela possibilidade de tratar os problemas gerais de transporte de forma razoável e eficiente, com o uso de bons algoritmos, como os do tipo Dual-Simplex, Out-of-Kilter, [Ford & Fulkerson 62], [Dinic 70], dentre outros.

O SisGRAFO utiliza o algoritmo Primal-Dual Out-of-Kilter, [Bazaraa et al 90]. O primeiro refere-se aos problemas de transporte e transporte com transbordo, e o segundo refere-se aos problemas de fluxo máximo e de fluxo de custo mínimo. Uma visualização animada dos resultados de fluxos foi inserida no SisGRAFO com o objetivo de dar ao usuário um melhor sentimento sobre o envio de fluxo e circulação na rede de trabalho. A figura II.27 mostra o resultado de uma animação sobre fluxo em redes de custo mínimo.

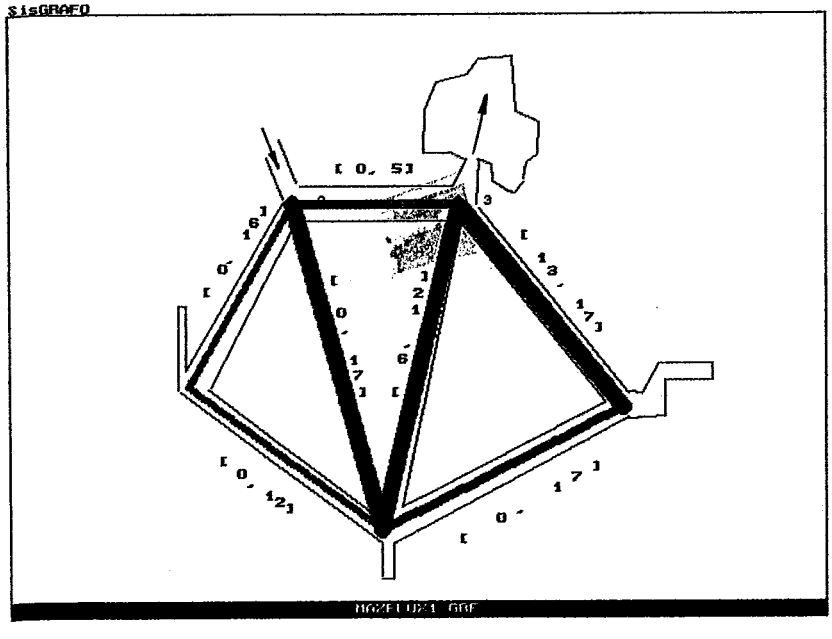


Figura II.27: Exemplo de aplicação (animada) de Fluxo em Redes de Custo Mínimo.

4. Localização

Dado um conjunto de clientes com demandas e custos relativos de serviço a cada supridor e um conjunto de locais candidatos a suprirem estas deman-

das com restrições de serviço associadas. Deseja-se saber quais serão os k-Supridores que permitirão atender à demanda total dos clientes e com custo de serviço mínimo.

Este é um problema que se aplica a uma grande variedade de situações reais, como na localização de aterros e incineradores, escolas, postos de bombeiros, postos de ambulância, postos de atendimento à população, postos de saúde, aeroportos, fábricas, etc. Além disto, é de grande complexidade, [Larson & Odoni 81].

Para resolvê-lo, o SisGRAFO utiliza dois modelos de localização. Um com alocação de k-depósitos pelo método do custo linear (DLOC), [Eilon et al 71]; e outro considerando que k-centros já estão disponíveis, e possuem custos limitados a faixas de produção por unidade de produto (ALOC).

O problema DLOC foi adaptado para resolver problemas de localização com depósitos fixos e restrições sobre áreas específicas, tendo sido aplicado em uma situação de planejamento real na localização de uma estação de transferência de lixo na cidade de Fortaleza, figura II.28, [Negreiros 96].

5. Rotas

Os problemas clássicos de geração de rotas têm obtido grande atenção nos últimos anos, em virtude de sua proximidade com a realidade em diversas áreas. Duas famílias de problemas de rotas existem: Caixeiro Viajante (Roteamento em Nós) e do Carteiro Chinês (Roteamento em Arcos), [Lenstra & Rinooy Kan 81].

Os problemas de rotas da família do problema do caixeiro viajante, são: 1-caixeiro viajante, caminho hamiltoniano, caixeiro viajante min-max (gargalo), m-caixeiro viajante, caixeiro viajante generalizado, caixeiro coletor de valores, viajante de compras, dentre vários outros que surgem como subproblemas dos citados, abordados neste texto nos capítulos III e IV.

Extensões maiores do problema do caixeiro, são os problemas de Roteamento de Veículos (em nós), e o Problema *Shopping* ou de Compras, discutidos a seguir neste texto. Tais problemas determinam demandas nas cidades e um veículo que deverá sair para satisfazê-la, ou até mesmo suas necessidades de compra. Estes problemas se aplicam geralmente na distribuição de refrigerantes e cervejas, leite, rotas de ônibus escolares e de trabalhadores, dentre muitos outros.

Já os problemas advindos do carteiro chinês, [Kwan Mei-Ko 62], são aqueles em que as rotas devem passar por um subconjunto conexo determinado de ligações a partir de um vértice origem de um grafo. Desta família temos: Carteiro chinês (não-direcionado, direcionado, misto, rural, *íngreme*), carteiro capacitado, *Stacker-Crane*, problema capacitado em arcos, dentre outros, discutidos a seguir neste texto. Suas aplicações estão geralmente relacionadas à

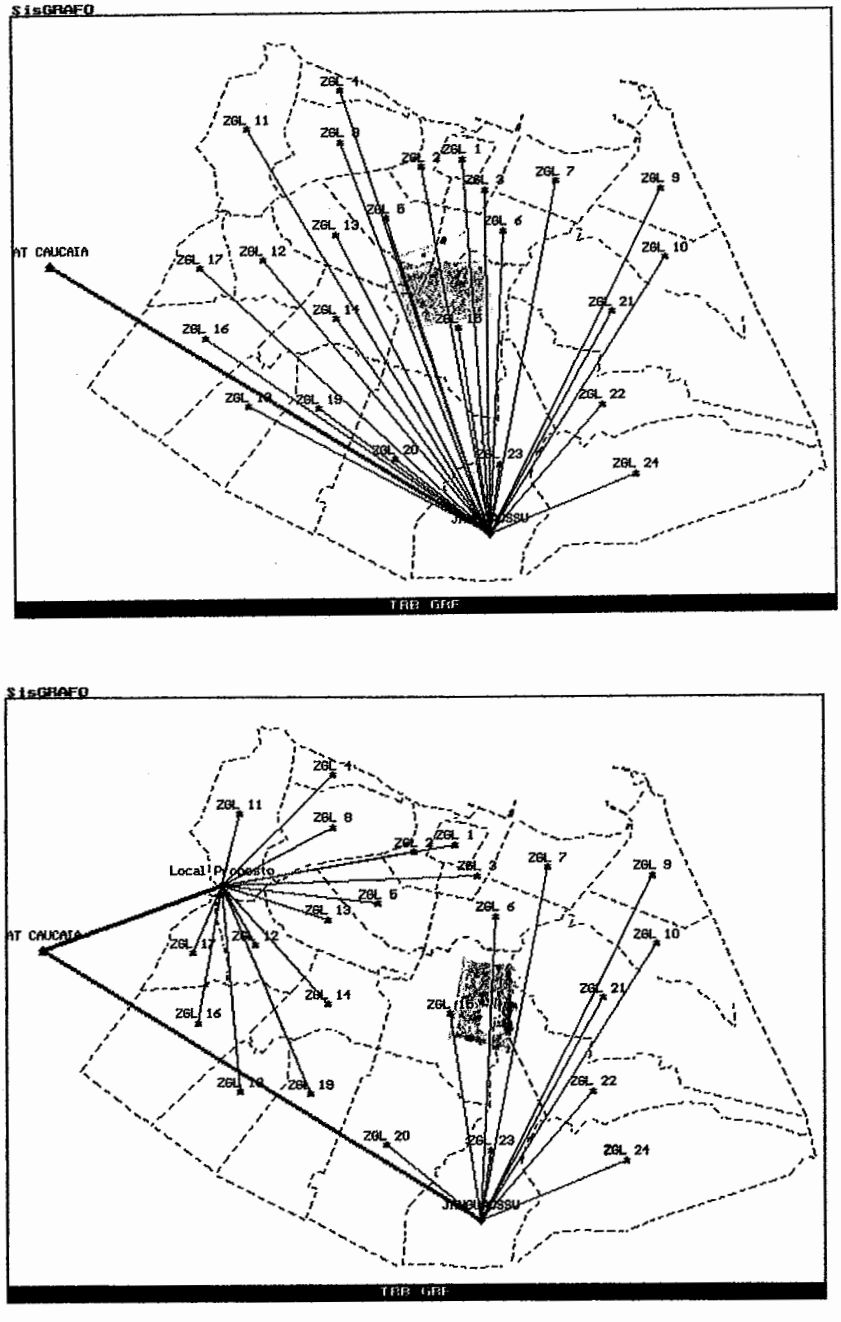


Figura II.28: Situação anterior de transferência e proposta de melhoria do sistema de coleta de lixo de Fortaleza usando o algoritmo DLOC implementado no SisGRAFO.

coleta de lixo, distribuição de gás engarrafado, pulverização de ruas, correios, dentre outras.

Em sua maioria, os problemas de rotas são considerados como $\mathcal{NP}$- Árduos, fato que justifica o uso de heurísticas para resolvê-los, na obtenção de soluções sub-ótimas ou mesmo ótimas (Porém sem garantia de otimalidade). Heurísticas clássicas são largamente utilizadas nos problemas do tipo caixeiro viajante. Métodos exatos do tipo *Branch & Bound* são também muito usados, mas se rescentem pelo reduzido número de vértices que ainda conseguem tratar.

O SisGRAFO possui um conjunto de algoritmos para o tratamento de rotas da família de problemas do caixeiro viajante e do carteiro chinês, cujos problemas tratados e respectivos algoritmos utilizados são os seguintes:

(a) *1*-Caixeiro, Caminho Hamiltoniano e *m*-Caixeiro, figuras II.29 e II.30
   - Branch & Bound, [Syslo et al 83] adaptado para o caso simétrico;
   - Vizinho Mais Distante e Mais Próximo (VMD, VMP);
   - 2-Otimal (2OPT) [Syslo et al 83] adaptado para o caso assimétrico;
   - 3-Otimal (3OPT), [Syslo et al 83];
   - VMD/MP+2OPT+3OPT;
   - OR-Otimal, [Nurmi 91].

(b) Compras
   - Caixeiro Aquisitor, ou Viajante de Compras - (*Purchaser*);
   - Problema de Compras - (*Shopping*).

(c) Percurso em Nós (adaptado para multi-depósito e um destino fixo), figura II.31
   - Clarke & Write Paralelo
   - Clarke & Write Sequencial

(d) Percurso de Veículos Multi-compartimentados (multi-depósito)
   - Paralela
   - Sequencial

(e) Percurso em Arcos (multi-depósito, usa animação)
   - Carteiro Chinês (simétrico, direcionado, misto, *stacker-crane*)
   - Carteiro Rural
   - Percurso em Arcos com Demanda

(f) Percurso Geral (Arcos e/ou Nós, usa animação) figura II.32
   - Carteiro Chinês (simétrico, direcionado, misto, *stacker-crane*)
   - Carteiro Rural
   - Percurso em arcos e/ou nós com Demanda

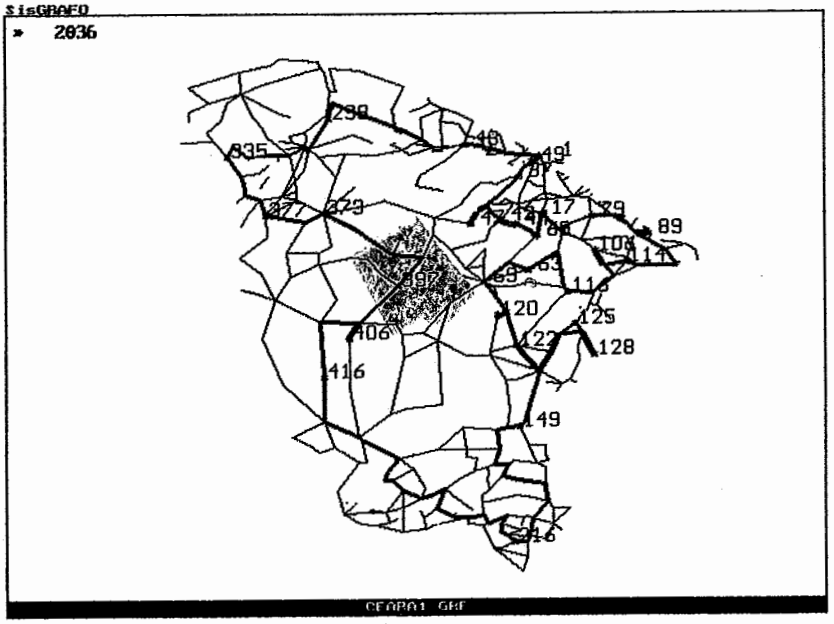


Figura II.29: Exemplo de Solução do 1-Caixeiro

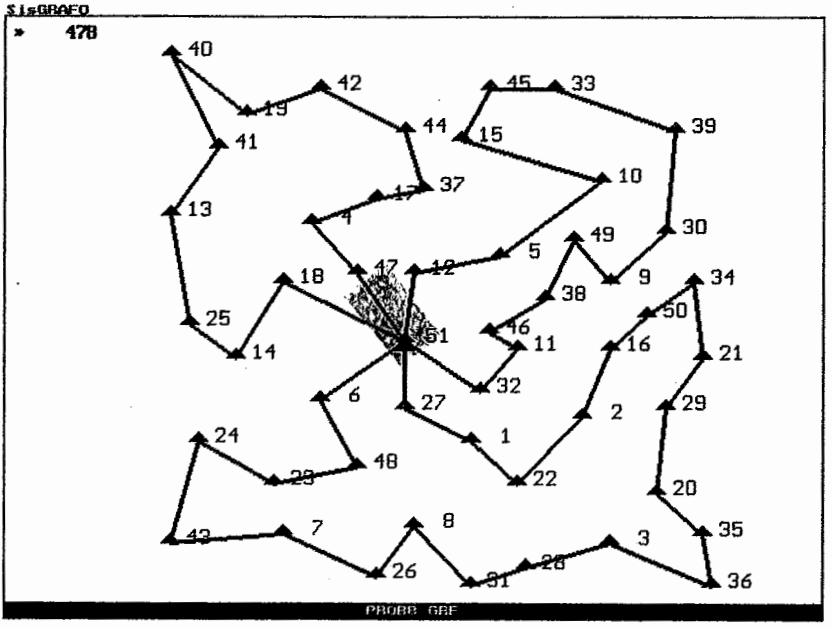


Figura II.30: Exemplo de Solução do m-Caixeiro.

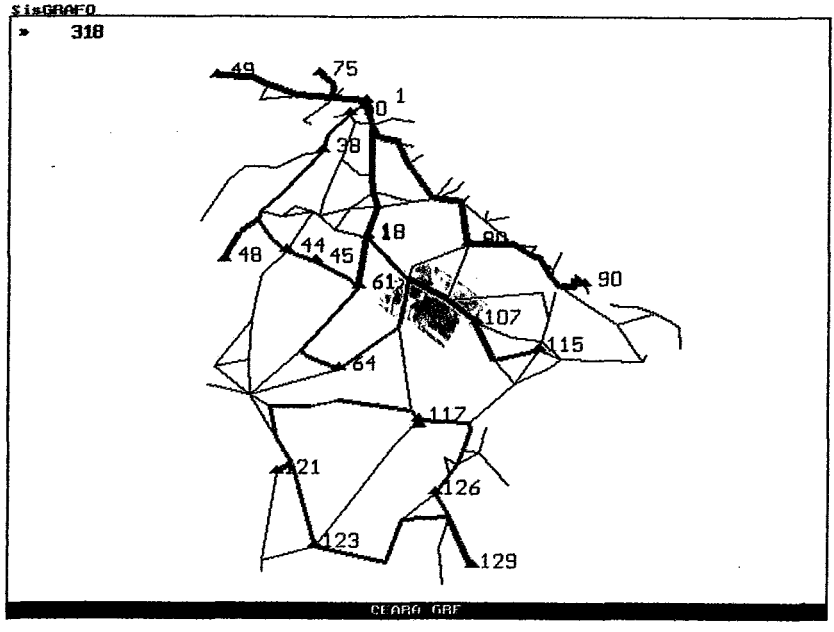


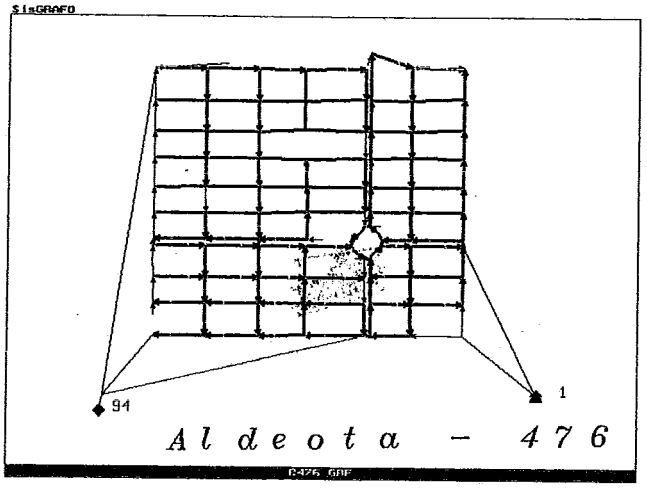
Figura II.31: Exemplo de Solução do Problema de Percurso de Veículos.

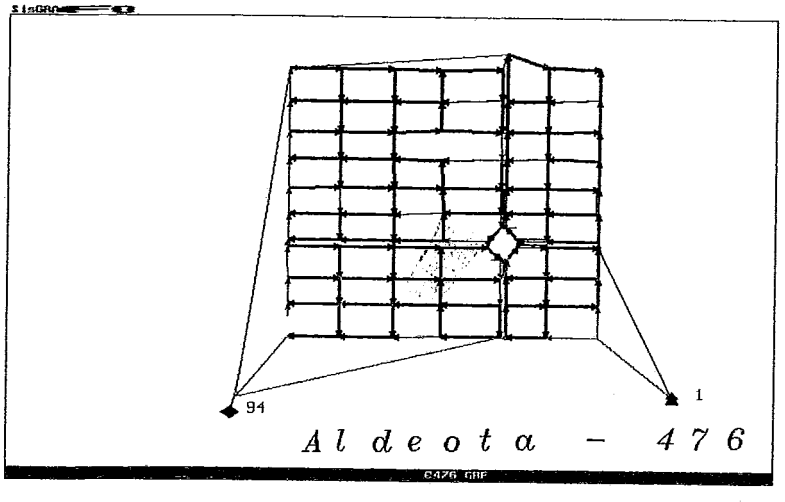
6. Atribuição

São também considerados como problemas de atribuição aqueles referentes à formação de agrupamentos simples e complexos.

Os problemas enfocados são o problema de:

- Atribuição (*assignment*), onde são dados vértices em uma rede e deseja-se atribuí-los entre si, ou entre dois tipos distintos de vértices (origem a clientes, ou homens a máquinas), com o custo total mínimo de atribuição (atribuição soma linear mínima);
- Emparelhamento (1-*Matching*), onde são dados vértices de uma rede, o problema refere-se a emparelhar (formar casais) os vértices com o menor custo global no emparelhamento.
- Agrupamento Capacitado (PAC), onde são dados vértices com demanda dispersamente colocados em uma rede valorada, onde se deseja formar um dado conjunto valorado de grupos bem definidos (homogêneos ou não), com o menor custo global no agrupamento dos vértices nos grupos. Na figura II.33 tem-se um exemplo de uma solução de um instância do PAC, onde tem-se 101 vértices com demanda unitária e são formados 4 grupos com demanda máxima de 26 unidades.





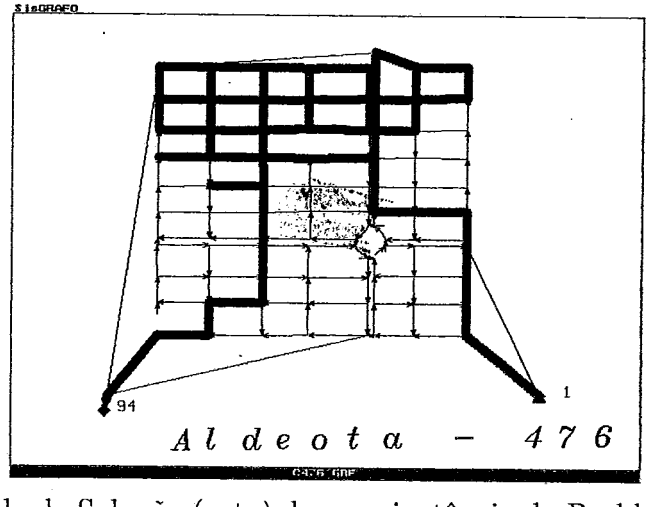


Figura II.32: Exemplo de Solução (rota) de uma instância do Problema de Percurso Geral.

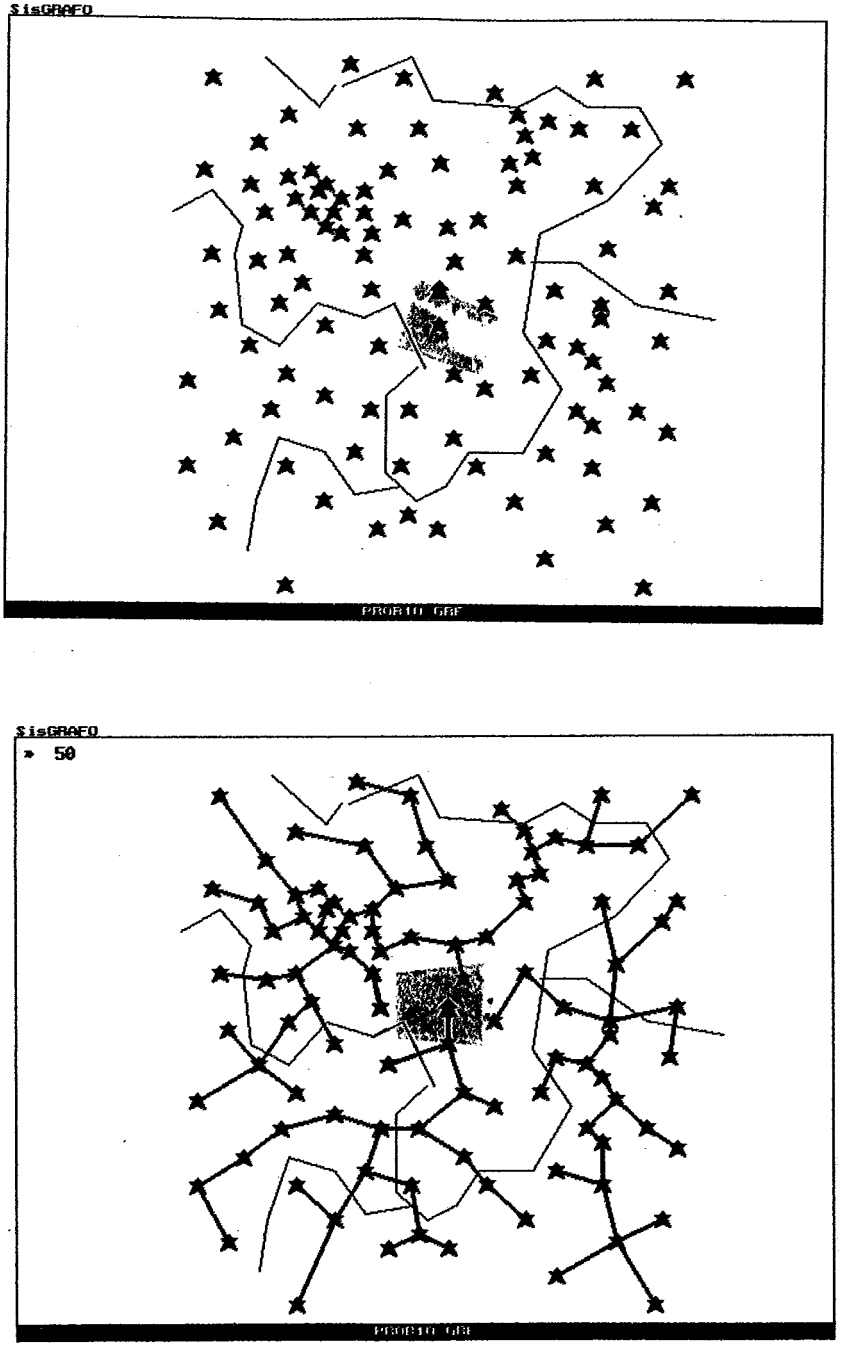


Figura II.33: Exemplo de Solução de uma instância do Problema de Agrupamento Capacitado.

Estes problemas são muito úteis para as análises de sequenciamento de tarefas e planejamento operacional. No SisGRAFO são utilizados os algoritmos propostos por [Burkard & Derigs 80] para o problema de atribuição e 1-*Matching*, e de [Mulvey & Beck 84] para o Problema de Agrupamento Capacitado.

Outros problemas serão tratados futuramente por versões mais avançadas do SisGRAFO, a seguir descreve-se as implementações previstas:

1. Árvores Geradoras, pretende-se tratar o problema de Steiner Euclidiano e de Manhattan, além do problema de k-árvores com grau máximo e k árvores capacitadas;

2. Caminhos, pretende-se as aplicações a problemas de k-caminhos com restrições de deslocamento;

3. Fluxos, pretende-se estender a análise aos casos de múltiplas comodidades, ganhos e custos constantes;

4. Localização, pretende-se considerar o problema da localização capacitada no contexto ALOC, e inserir condições de localização com múltiplos produtos para a versão DLOC;

5. Caixeiro Viajante, pretende-se considerar o problema min-max (gargalo), do caixeiro generalizado e do coletor de valores, além de uma implementação exata mais expressiva do 1-caixeiro para os casos simétrico e assimétrico;

6. Problemas de Percurso de Veículos, implementação dos métodos com janelas de tempo, percurso com frota mista e percurso periódico;

7. Atribuição, pretende-se inserir o Problema de Atribuição Generalizado e o Problema de Atribuição Quadrático.

II.4.1 Perspectivas do SisGRAFO

A versão 1.0 do SisGRAFO considera uma arquitetura fechada, baseada em programação estruturada, usando grafos apenas como modelos para instâncias. Um editor de instâncias foi elaborado, porém um editor de esquemas não existe, uma vez que esta tarefa foi exclusivamente atribuida ao projetista do sistema. No SisGRAFO ainda não existe uma gramática de grafos aberta ao usuário (modelador) no que concerne a produções para manipulação de soluções de problemas específicos.

O critério de se tomar a partir de um grafo dado um conjunto de instâncias particulares que se adequem a problemas diversos é um pouco diferente

e paradigmático em relação ao que foi proposto por Jones. Porém, mesmo o trabalho de Jones esbarra na etapa de solução de problemas, pois nem sempre funções matemáticas (macros) agindo sobre os atributos de vértices e ligações conseguem resolver diretamente problemas não convencionais, diferentes daqueles sobre Árvores de Decisão e problemas de Caminho Crítico considerado por Jones. O tratamento de uma instância por um modelo pré-concebido é algo aceitável e direto. No SisGRAFO, um editor de problemas poderia surgir como uma visão superior ao editor de esquemas, onde este incorporaria as idéias condicionais, *loops*, e desvios de uma linguagem de programação em modo de interpretação de comandos.

Mudanças estruturais em atributos são também um ponto forte, uma vez que se pode referenciá-las a um conjunto de funções matemáticas (macros) as quais ajudarão na composição de resultados e na extensão do uso do sistema a outros softwares tais como planilhas eletrônicas e editores de texto. Um de soluções deve figurar como um elemento fundamental a ser inserido, já com produções definidas para os diversos problemas incorporados ao sistema. Também prevê-se uma mudança para ambientes multitarefas, onde o usuário poderá rodar vários problemas distintos simultaneamente para o mesmo grafo, já estou trabalhando nisto com afinco.

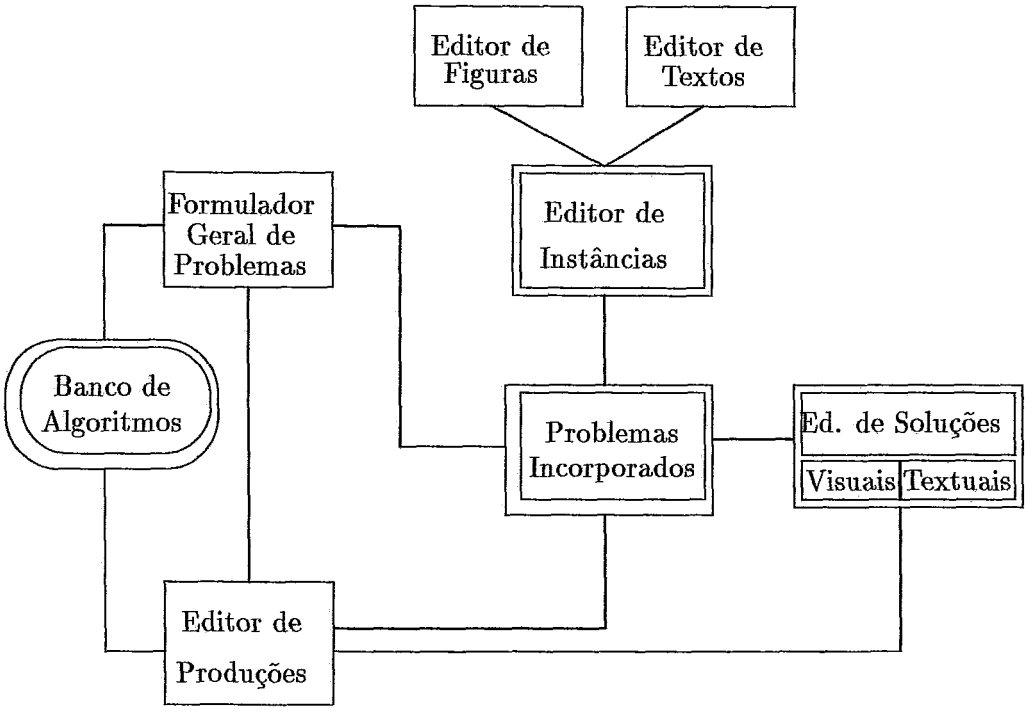


Figura II.34: Diagrama das perspectivas do SisGRAFO.

A figura II.34 mostra a minha perspectiva sobre este sistema, uma vez que é inevitável a sua expansão e continuidade.

Capítulo III

O Problema do Caixeiro Viajante

O Problema do Caixeiro Viajante (PCV) é um problema bastante conhecido e estudado pela literatura, considera que *um vendedor, iniciando da sua cidade irá visitar exatamente uma vez uma lista de cidades e então retornará à sua casa de tal modo que para ele é plausível definir uma ordem de visita das cidades de modo que a distância total percorrida seja a menor possível.*

Como se pode ver apesar de tentar minimizar a distância total da viagem do caixeiro, o PCV é um problema do tipo combinatorial, pois trata das diversas sequências de visitação de um conjunto de cidades, de tal modo que um determinado conjunto destas sequências atingirá a melhor distância possível.

A importância do PCV vem de um conjunto de aplicações relacionadas a Problemas de Otimização Combinatória, algumas delas são descritas a seguir, [Laporte 92], [Lawler et al 85] e [Miller & Pekny 91].

1. **Roteamento de Veículos**, que é o problema de determinar para uma frota de veículos os clientes que serão servidos por um dado veículo, de tal modo que a frota atenda a todos os clientes. As restrições do problema em geral consideram que os clientes têm demandas distintas e os veículos têm uma capacidade limitada de atendimento, tempo máximo de atendimento, distância máxima percorrida, dentre outras.
2. **Conecção de Computadores**, quando no projeto de computadores ou outros sistemas digitais. Alguns destes sistemas podem ser descritos como módulos com pinos que os fixam. Neles é sempre necessário ligar estes pinos através de fios, de tal modo que exatamente dois fios são ligados a cada pino e o comprimento total de fio deve ser minimizado.
3. **Corte de Papel de Parede**, onde supondo-se que se deseja cortar n folhas de papel de parede a partir de um único rolo de papel bem longo, sobre o qual um dado padrão de corte de comprimento unitário se repete.

A folha i inicia na posição a_i e termina na posição b_i sobre o padrão. (Portanto, $b_i = a_i + l_i (mod\ 1)$ onde o comprimento da folha i é l_i unidades do padrão). Assim, o corte de uma folha j logo a seguir a folha i resulta numa perda de:

$$c_{ij} = \begin{cases} a_j - b_i, & \text{se } b_i \leq a_j, \\ 1 + a_j - b_i, & \text{se } b_i > a_j. \end{cases} \tag{III.1}$$

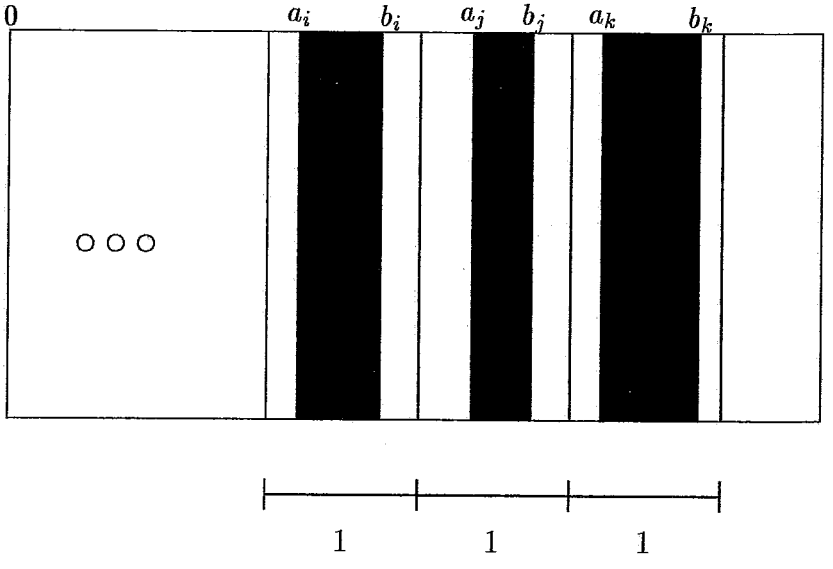
O objetivo é de ordenar as n folhas de modo a minimizar a perda total. Com o objetivo de colocar o problema como um PCV, define-se uma folha *controle* $n+1$ com $c_{i,n+1} = 0$ e $c_{n+1,j} = 0$ para todo $i, j = 1, \ldots, n$. Alternativamente, define-se b_{n+1} como o final da posição do rolo e o início do corte e admite-se que após a última folha, um corte final pode ser feito para restaurar a linha de sua posição original. Então $a_{n+1} = b_{n+1}$ e c_{ij} pode ser definido como em III.1 para todo $i, j = 1, \ldots, n+1$. A figura III.1 mostra a idéia de resequenciamento a partir de uma configuração inicial.

4. **Perfuração de Placas**, em alguns contextos de manufatura e na construção de placas de circuito impresso, necessita-se furar buracos em quadros ou em placas de metal. O problema consiste em determinar o menor tempo na sequência de perfuração. Em geral, estes problemas acontecem onde existem um grande número de furos a fazer, em sistemas automáticos com grande volume de placas a perfurar.

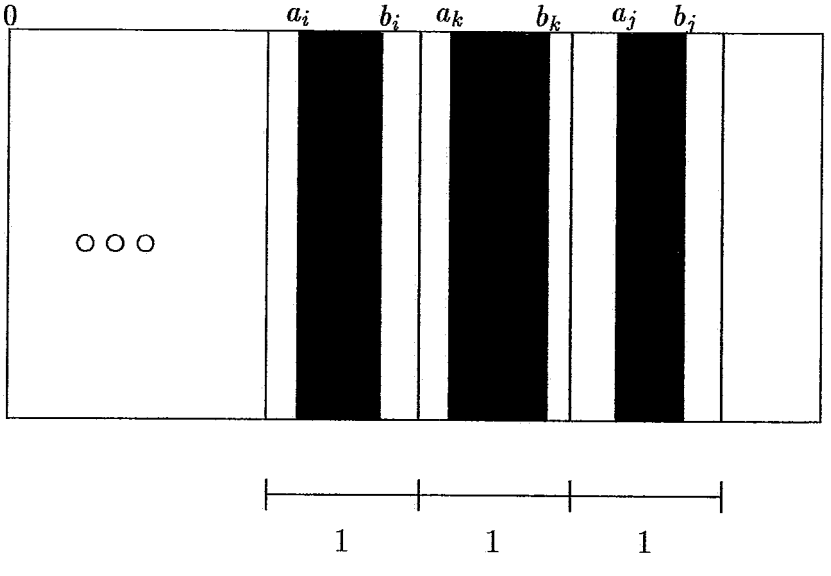
5. **Sequenciamento do Trabalho**, onde supõe-se que n tarefas devem ser executadas sequencialmente em uma única máquina e que c_{ij} é o tempo de mudança caso o trabalho j seja executado logo após o trabalho i. Este problema pode ser analisado de forma similar ao problema de cortes de papel de parede.

6. **Desenho de Placas de Dardo**, onde as placas de Dardo são alvos circulares com círculos concêntricos, e vinte setores rotulados de 1 a 20. No jogo os competidores *atiram* dardos na pontuação alvo sobre a placa. Na sua versão mais comum, o objetivo é reduzir um valor inicial de 301 a zero pela subtração dos escores obtidos. O jogo premia a presteza nas jogadas do dardo e quase sempre é mais importante atingir um objetivo pessoal do que meramente registrar um escore maior. Um objetivo natural no projeto de placas de dardo é pois posicionar os 20 números ao redor da placa de modo a maximizar o risco do jogador. Para jogadores de pouca pontaria, é razoável admitir que o setor que ele atinge é sempre o setor alvo ou seu vizinho.

   Seja $\pi = (\pi(1), \ldots, \pi(20))$ qualquer permutação dos números de 1 a 20. Segue-se que, $\pi(k)$ deve ser considerado como $\pi(k\ mod\ 20)$ sempre que $k < 1$ ou $k > 20$. Considerando-se que um jogador atingindo em $\pi(k)$ e batendo $\pi(k \pm 1)$ com probabilidade p, e $\pi(k)$ com probabilidade $1 - 2p$.



Configuração Inicial de uma Sequência de Padrões.



Configuração melhorada de uma Sequência de Padrões.

Figura III.1: Resequenciamento no posicionamento dos padrões para redução de perdas em um rolo de papel de parede.

Para este jogador, o desvio esperado para o escore pretendido é igual a $p[\pi(k-1)-\pi(k)]+p[\pi(k+1)-\pi(k)]$. Um objetivo possível é maximizar a soma esperada dos desvios quadráticos, ou seja: $\sum_{k=1}^{20}\{p[\pi(k-1)-\pi(k)]^2 + p[\pi(k+1)-\pi(k)]^2\}$. Uma vez que p é uma constante, isto é equivalente a resolver o PCV com $c_{ij} = (i-j)^2$.

7. **Cristalografia**, nesta área alguns experimentos consistem em tomar um grande número de medidas de intensidade de raios-X sobre um cristal através de um detector. Cada medida requer que uma amostra do cristal seja montada sobre um aparato e que o detector seja posicionado apropriadamente. A ordem na qual as diversas medidas são feitas sobre um dado cristal pode ser vista como uma solução de um PCV. Na prática, estes problemas são de grande porte e a obtenção de boas soluções do PCV pode reduzir de forma considerável o tempo necessário à realização de todas as medidas.
8. **Outras Aplicações** tais como no escalonamento de processos químicos, alocação de padrões na indústria de vidro, na vistoria de motores com turbinas a gás dentre outras podem surgir.

Este capítulo trata analiticamente o ponto de vista da Programação Matemática o PCV, e os principais algoritmos exatos e heurísticos desenvolvidos para a resolução de instâncias de larga escala do PCV.

III.1 Definição do PCV

Seja $G = (V, L)$ um grafo onde V é um conjunto formado por n vértices ou nodos. L é um conjunto de ligações formado por elos ou arcos, e seja $C=(c_{ij})$ a matriz de custos (ou de distâncias) associada a L. O PCV consiste em determinar o circuito de distância mínima passando por cada vértice somente uma vez. Tal circuito é conhecido como uma *rota* ou *Circuito Hamiltoniano* (ou *ciclo*).

Em algumas aplicações C pode ser admitida como uma matriz de custos ou de tempos de percurso. É importante porém fazer distinção clara entre os casos onde C é *simétrica*, ou seja, quando $c_{ij} = c_{ji}$ para todo $i, j \in V$, e o caso onde é *assimétrica*. Também diz-se que C satisfaz à *desigualdade triangular* se e somente se $c_{ij} + c_{jk} \geq c_{ik}$ para todo $i, j, k \in V$. Isto ocorre em problemas Euclidianos, ou seja, quando V é um conjunto de pontos em $\Re^2$ por exemplo e c_{ij} é um segmento de reta entre i e j.

Considerando a complexidade do PCV, é necessário considerar inicialmente o Problema do Circuito Hamiltoniano (PCH) onde, dado um grafo $G = (V, L)$ deseja-se saber se G contém um circuito hamiltoniano -, ou seja: um percurso

em G que passa por todos os seus vértices a partir de um vértice origem e a ele retornando.

Sabe-se que o PCH é NP-Completo, [Garey & Johnson 79]. Já [Laporte 92] mostra que o PCV é NP-Árduo pelo uso da seguinte transformação. Dada qualquer instância do PCH relativa a um grafo $G^h = (V^h, L^h)$ com o conjunto de vértices $V^h = \{1, \ldots, n\}$ e o conjunto de ligações $A^h = \{(i,j) : i,j = 1, \ldots, n, i \neq j\}$ e $c_{ij} = 1$ se $(i,j) \in L^h$ e $c_{ij} = \infty$, caso contrário. Então G contém um circuito hamiltoniano se e somente se o valor ótimo de uma instância do PCV é igual a n.

Alguns casos, porém podem ser resolvidos em tempo polinominal, entre eles temos:

1. Os PCVs onde $C = (c_{ij})$ é uma matriz triangular superior, ou seja, $c_{ij} = 0$ para todo $i \geq j$;

2. O problema do corte de papel de parede descrito no inicio deste capítulo;

3. Uma classe de problemas de sequenciamento de trabalhos;

4. Matrizes graduadas, onde diz-se que C é uma matriz graduada se $c_{ij} \leq c_{i,j+1}$ para todo i,j e suas colunas são *graduadas* se $c_{ij} \geq c_{i+1,j}$ para todo i,j. Uma matriz é *duplamente graduada* se é graduada em ambas as suas linhas e acima de suas colunas.

5. Rotas Piramidais, uma rota τ sobre n cidades 1,2, ...,n é *piramidal*, se τ é da forma $(1, i_1, \ldots, i_r, n, j_1, \ldots, j_{n-r-2})$, onde $i_1 < i_2 < \ldots < i_r$ e $j_1 > j_2 > \ldots > j_{n-r-2}$, figura III.2.

III.2 Formulações Matemáticas

Assume-se por simplicidade que C é *simétrica* e que $c_{ii} = \infty$ para todo i. O problema, como já sabemos, é formar uma viagem que passa por todos os vértices iniciando e terminando na origem, vértice 1, que dá a distância total mínima da rota.

Considera-se a variável de decisão como sendo,

$$x_{ij} = \begin{cases} 1, & \text{se a ligação } ij \text{ está na viagem} \\ 0, & \text{caso contrário.} \end{cases}$$

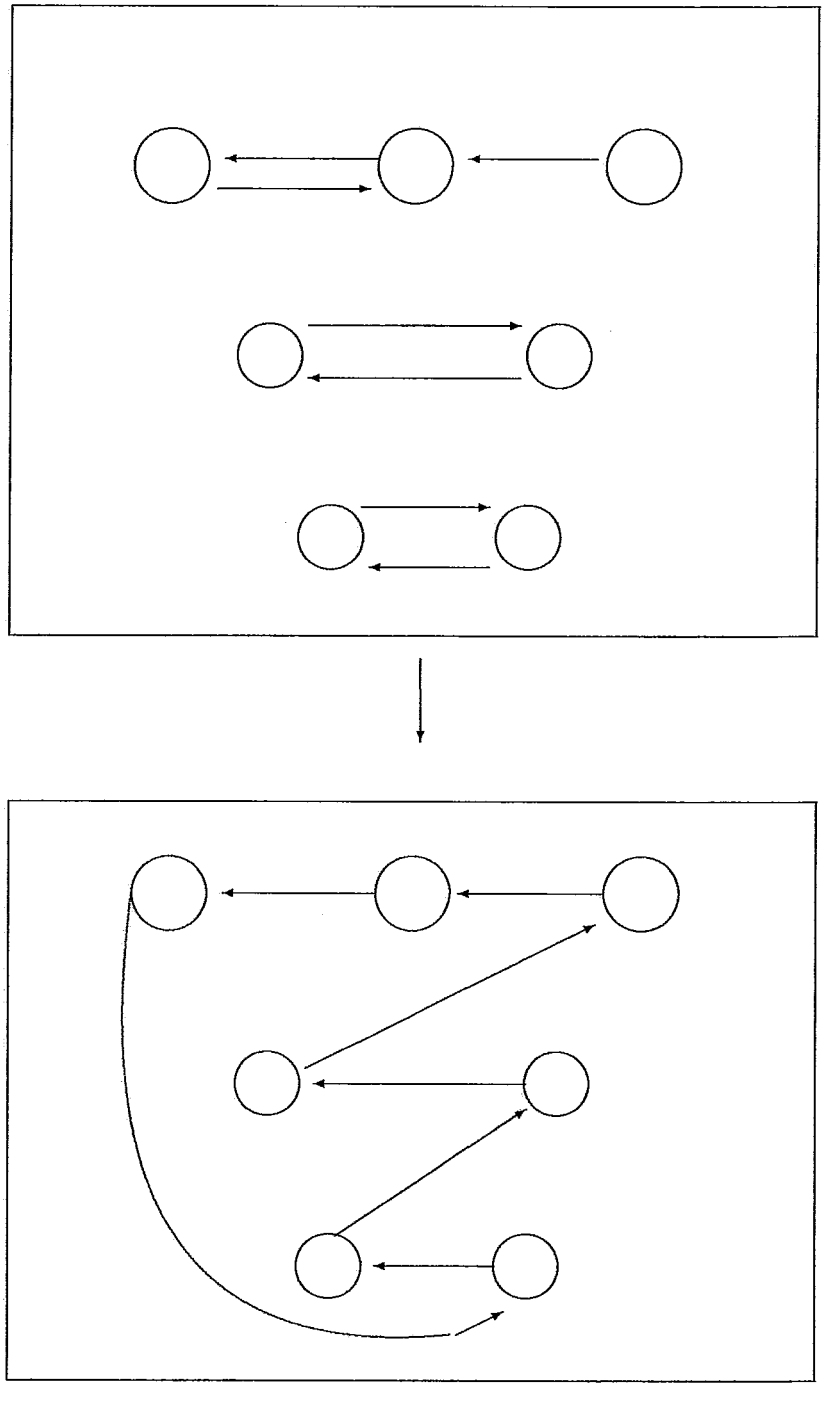


Figura III.2: Uma atribuição ótima de ϕ e uma Rota Piramidal obtida de ϕ.

III.2.1 Formulação Baseada em Atribuição

Esta formulação, [Bodin et al 83], seleciona a matriz $X = (x_{ij})$ de variáveis de decisão tal que exatamente uma ligação ij está direcionada a cada nó j. Nota-se porém que as necessidades da atribuição, por elas mesmas, não levam a que a matriz X corresponda a uma viagem. Ao invés, a atribuição resulta em subciclos como pode ser visto na figura III.3.

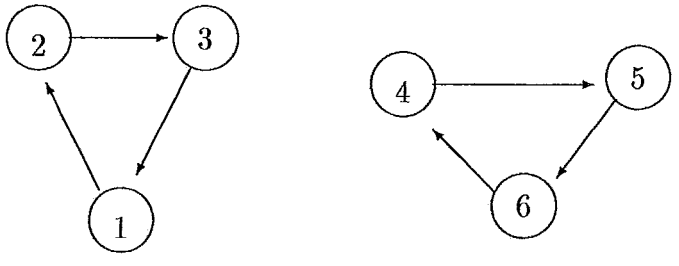


Figura III.3: Formação de subciclos em um problema de atribuição simples.

Com o objetivo de eliminar a possibilidade de geração de subciclos, restrições adicionais são impostas sobre a escolha dos arcos da matriz X que conduz à formulação:

$$\textbf{(PCVa)} \quad \text{Minimizar} \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}\, x_{ij} \tag{III.2}$$

sujeito a,

$$\sum_{i=1}^{n} x_{ij} = b_j = 1 \ (j = 1, \ldots, n) \tag{III.3}$$

$$\sum_{j=1}^{n} x_{ij} = a_i = 1 \ (i = 1, \ldots, n) \tag{III.4}$$

$$X = (x_{ij}) \in S \tag{III.5}$$

$$x_{ij} \in \{0, 1\} \ (i, j = 1, \ldots, n) \tag{III.6}$$

O conjunto S pode ser formado por quaisquer restrições que proibam soluções de subciclos satisfazendo as regiões de atribuição III.3, III.4 e III.6. Tais restrições são chamadas de restrições de quebra de subciclos. Algumas escolhas para S podem ser:

1. $S = \{x_{ij} : \sum_{i \in Q} \sum_{j \notin Q} x_{ij} \geq 1$, para todo subconjunto próprio não vazio Q de $V\}$
2. $S = \{x_{ij} : \sum_{i \in R} \sum_{j \notin R} x_{ij} \leq |R| - 1$, para todo subconjunto não vazio R de $\{2,3, \ldots, n\}$ $\}$
3. $S = \{x_{ij} : y_i - y_j + n x_{ij} \leq n - 1$, para $2 \leq i \neq j \leq n$, para algum número real $y_i\}$

Nota-se que S contém aproximadamente 2^n restrições de quebra de subciclos em 1 e 2, enquanto possui $n^2 - 3n + 2$ na formulação 3. Nota-se também que as restrições III.3, III.4 e III.6 estão satisfeitas para o problema da figura III.3, mas não a restrição III.5. Ou seja, ela não representa um caminho hamiltoniano.

O primeiro conjunto de restrições de quebra de subciclos, implicam que os arcos selecionados em X não contêm ciclo, desde que um ciclo sobre os nós R devem conter $|R|$ ligações. Isto inclui a solução da figura III.3 quando $R = \{4, 5, 6\}$.

O terceiro conjunto de restrições para quebra de subciclos é mais complicado, adicionando as restrições 3 às ligações (4,5), (5,6) e (6,4) da figura III.3, permite-se que $3n \leq 3(n-1)$, que é uma contradição. Uma contradição idêntica surge quando a matriz X contém uma solução com subciclo. Portanto cada conjunto de restrições 1, 2 e 3 previne a formação de subrotas.

É fácil ver quando as rotas são viáveis em cada caso. As restrições 1 e 2 não apresentam dificuldades. Na restrição 3 seja,

$$y_i = \begin{cases} t, & \text{Se o nó } i \text{ é visitado no } t\text{-ésimo passo numa rota} \\ 0, & \text{Caso Contrário.} \end{cases}$$

Então, cada restrição (3) associada a um arco em uma rota significa que $t - (t + 1) + n \leq n - 1$. Outras restrições em (3) revelam apenas que $y_i - y_j \leq n - 1$ sempre que $y_i \leq n$ e $y_j \leq 1$.

III.2.2 Formulação Baseada em Fluxo

[Gavish & Graves 78] introduziram variáveis de fluxo y_{ij} e assumiram que $(n-1)$ unidades de fluxo é suprida pela origem, nó 1, e que todos os outros vértices da rede necessitam de apenas uma unidade de fluxo. A formulação pode ser colocada como segue.

$$\textbf{(PCVf)} \quad \text{Minimizar} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}\, x_{ij} \qquad \text{(III.7)}$$

sujeito a,

$$\sum_{i=1}^{n} x_{ij} = 1 \ (j = 1, \ldots, n) \tag{III.8}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \ (i = 1, \ldots, n) \tag{III.9}$$

$$\sum_{j=1}^{n} y_{ij} - \sum_{j=2}^{n} y_{ji} = 1 \ (i = 2, \ldots, n) \tag{III.10}$$

$$y_{ij} \leq (n-1) \ x_{ij} \ (i = 2, \ldots, n, \ j = 1, \ldots, n) \tag{III.11}$$

$$x_{ij} \ \in \ \{0, 1\}, \ y_{ij} \geq 0 \ (i, j = 1, \ldots, n) \tag{III.12}$$

As restrições III.8 e III.9 são de atribuição como antes. As restrições para *forçar fluxo* III.11 faz com que o fluxo no arco y_{ij} sobre o arco (i, j) seja zero se ele não for selecionado na matriz de atribuição X. A formulação proibe a formação de subciclos, uma vez que se o vértice i recai sobre um subciclo que não contenha a origem, não poderia receber uma unidade de demanda da origem.

Esta formulação usa $O(n^2)$ variáveis binárias x_{ij}, $O(n^2)$ variáveis contínuas y_{ij} e $n^2 + 3n$ restrições. Como se pode notar, nesta formulação a variável y_{ij} descreve o fluxo de uma única comodidade do vértice origem a todos os outros vértices.

III.2.3 Formulação em Fluxo com Multi-Comodidades

[Wong 80] apresenta uma formulação para o PCV usando o fluxo de $2(n-1)$ comodidades, $Y^k = (y_{ij}^k)$, $k = 2, \ldots, n$ e $Z^k = (z_{ij}^k)$, $k = 2, \ldots, n$.

$$(\textbf{PCV}\text{mc-f}) \quad \text{Minimizar} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} \ x_{ij} \tag{III.13}$$

sujeito a,

$$\sum_{i=1}^{n} x_{ij} = 1 \ (j = 1, \ldots, n) \tag{III.14}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \ (i = 1, \ldots, n) \tag{III.15}$$

$$\sum_{j=1}^{n} (y_{ij}^k - y_{ji}^k) \ = \ \left\{ \begin{array}{ll} 1, & \text{Se } i = 1 \\ -1, & \text{Se } i = k \\ 0, & \text{Se } i \neq 1 \text{ e } k \end{array} \right\} \ k = 2, \ldots, n \tag{III.16}$$

$$\sum_{j=1}^{n}(z_{ij}^k - z_{ji}^k) = \left\{ \begin{array}{ll} -1, & \text{Se } i = 1 \\ 1, & \text{Se } i = k \\ 0, & \text{Se } i \neq 1 \text{ e } k \end{array} \right\} \; k = 2, \ldots, n \qquad \text{(III.17)}$$

$$y_{ij}^k \leq x_{ij}, \quad z_{ij}^k \leq x_{ij} \;\; \forall\, i, j, k \qquad \text{(III.18)}$$

$$y_{ij}^k \geq 0, \quad z_{ij}^k \geq 0 \;\forall\, i, j, k \qquad \text{(III.19)}$$

$$x_{ij} \in \{0, 1\} \; (i, j = 1, \ldots, n) \qquad \text{(III.20)}$$

As restrições III.16 e III.17 garantem que uma unidade da comodidade Y^k viaja do vértice 1 (fonte de comodidade Y^k) ao vértice k (o sumidouro da comodidade Y^k) enquanto que uma unidade de comodidade Z^k atravessa do vértice k ao vértice 1. Isto implica que o subgrafo que suporta o fluxo é f-conexo, ou seja, há uma caminho sobre este subgrafo que vai de todo vértice a qualquer outro vértice (passando pela fonte). Esta formulação necessita de $O(n^2)$ variáveis binárias, $O(n^3)$ variáveis contínuas e $2(n^3+n^2+n)$ restrições. Somente as variáveis x_{ij} estão sujeitas à integralidade desde que y_{ij}^k e z_{ij}^k são variáveis necessariamente inteiras sempre que as x_{ij} também forem inteiras, devido à matriz de custos para as comodidades Y^k e Z^k serem totalmente unimodulares (as x_{ij}'s correspondentes às capacidades sobre os arcos). [Wong 80] também demonstrou que a solução de seu modelo é equivalente à solução proposta por [Dantzig et al 54].

Uma vez que o número de restrições de quebra de subrotas cresce exponencialmente no modelo proposto por [Dantzig et al 54] mas apenas de forma polinomial no modelo proposto por [Wong 80], o potencial de simplificação do problema é muito significativo. Nota-se porém que a formulação de [Wong 80] requer um número maior de variáveis.

III.2.4 Formulação em Fluxo com Duas-Comodidades

Com o objetivo de reduzir significativamente o número de variáveis do modelo proposto por [Wong 80], [Finke et al 84] propuseram uma formulação de fluxo com duas comodidades para o PCV. Considerando que o PCV sai do vértice 1 com $n-1$ unidades da comodidade Y e nenhuma da comodidade Z. Ele distribui uma unidade de Y e recolhe uma unidade de Z em cada vértice tal que o viajante mantém sempre sua viagem com $n-1$ unidades. O modelo então pode ser escrito como segue.

$$\textbf{(PCV2c-f)} \quad \text{Minimizar} \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} \frac{(y_{ij} + z_{ij})}{n-1} \qquad \text{(III.21)}$$

sujeito a,

$$\sum_{j=1}^{n}(y_{ij} - y_{ji}) = \begin{cases} n-1, & \text{Se } i = 1 \\ -1, & \text{Se } i \neq 1 \end{cases} \tag{III.22}$$

$$\sum_{j=1}^{n}(z_{ij} - z_{ji}) = \begin{cases} -(n-1), & \text{Se } i = 1 \\ 1, & \text{Se } i \neq 1 \end{cases} \tag{III.23}$$

$$\sum_{j=1}^{n}(y_{ij} - z_{ij}) = n-1, \ \forall\, i \tag{III.24}$$

$$y_{ij} + z_{ij} \in \{0, n-1\} \forall\, i,j \tag{III.25}$$

$$y_{ij} \geq 0, \ z_{ij} \geq 0 \ \ \forall\, i,j \tag{III.26}$$

As restrições III.22 e III.23 definem a conservação de fluxo para cada comodidade. As restrições III.24 e III.25 garantem que há exatamente um arco que suporta o fluxo combinado de $n-1$ unidades saindo de cada vértice. A estrutura do supridor e da demanda implica que há um caminho da fonte 1 a cada vértice j e um outro de cada vértice j a 1. Portanto, o grafo suporte está fortemente conexo assim como no caso da formulação multi-comodidade. A relaxação de Programação Linear (PL) da formulação anterior é obtida pela troca das restrições III.25 por $y_{ij} + z_{ij} \leq n-1, \ \forall\, i,j$. Estas restrições são redundantes devido a III.24 e podem ser eliminadas da relaxação de PL. O Problema Linear correspondente contém pois $3n$ restrições enquanto a formulação inteira contém $O(n^2)$ restrições adicionais.

[Lucena 86] generalizou a formulação de [Finke et al 84]. No caso, o caixeiro viajante inicia no vértice 1, distribui d_i unidades da comodidade Y e coleta d_i unidades da comodidade Z no vértice i, $i = 2, \ldots, n$. Em relação às equações de III.21 a III.26, d_i troca o valor 1 para $i \neq 1$ enquanto $\sum_{i=2}^{n} d_i$ troca os valores de $(n-1)$ em todo lugar. Esta formulação de duas comodidades tem o mesmo número de variáveis e restrições assim como a formulação proposta por [Finke et al 84].

As relações entre as formulações podem ser vistas em [Langevin, Soumis & Derosiers 89], que discutem com propriedade o desenvolvimento de formulações mais apuradas para o PCV, vindo do primeiro modelo [Dantzig et al 54] com $O(2^n)$ restrições para o modelo de [Finke et al 84] com $O(n^2)$ restrições.

III.3 Algoritmos para o PCV

A partir das formulações apresentadas na seção anterior, discute-se a seguir alguns métodos exatos e heurísticas para o encaminhamento das soluções de instâncias de grande porte do PCV.

III.3.1 Algoritmos Exatos

No início das avaliações utilizando o PCV, [Dantzig et al 54] descreveram como encontraram uma solução ótima para um PCV com 49 cidades. Iniciando por um limite superior determinado usando caracteres e marcas, eles relaxaram o PCV para um problema de atribuição ignorando temporariamente a eliminação de subciclos. Desigualdades, chamadas *cortes*, foram então introduzidas de forma a eliminar os subciclos e forçar que as variáveis de decisão x_{ij} viessem a se tornar inteiras. Quando a variação entre um limite superior e um limite inferior da relaxação era expressivamente pequeno, o tamanho do problema era reduzido através de um esquema de redução de uma variável fixa e o problema resultante era resolvido através de um esquema tipo *Branch-and-Bound* - B&B.

Apesar das instâncias do PCV serem resolvidas em [Dantzig et al 54] para tamanhos pequenos, e a implementação não ser totalmente automática, o trabalho é da maior importância pois considera os primeiros exemplos de técnicas importantes para o caso, tais como planos de corte, fixação de variáveis e B&B. Muitos trabalhos foram feitos na tentativa de formular melhores restrições de eliminação de subciclos e encontrar restrições de indução de faces, [Grötschel & Padberg 79a] e [Grötschel & Padberg 79b]. Este último trabalho permitiu um grande avanço no tamanho dos problemas que podem ser resolvidos exatamente. [Grötschel & Holland 91] desenvolveram uma solução baseada em geração de facetas do poliedro do PCV, que atinge a resolução de instâncias acima de 1000 cidades, porém necessitando de um tempo computacional bastante longo. Esta técnica é muito próxima à de [Dantzig et al 54] excetuando-se que os limites superiores e inferiores, e as facetas do poliedro (planos de corte), são gerados de forma algorítmica.

Uma outra técnica bastante utilizada na resolução de problemas simétricos do PCV refere-se ao uso da relaxação *1-tree* com uma função objetivo Lagrangeana, e então usa-se *Branch and Bound.* Esta técnica foi inicialmente explorada por [Held & Karp 70] e [Held & Karp 71], e mais tarde melhorada por [Volgenant & Jonker 82], [Gavish & Srikanth 86] e mais recentemente por [Carpaneto et al 89].

Já para o caso da resolução de problemas assimétricos, [Miller & Pekny 91] propuseram um novo algoritmo do tipo *Branch and Bound* bastante eficiente baseado na relaxação do Problema de Atribuição (PA). Eles consideram basicamente o problema dual do PA.

$$\textbf{(PAD)}\ \text{Maximizar} \sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j \qquad \text{(III.27)}$$

Sujeito a,

$$c_{ij} - u_i - v_j \geq 0 \tag{III.28}$$
$$i, j = 1, \ldots, n, \; i \neq j \tag{III.29}$$

Chamando por z^*(PCV) o valor da solução ótima do PCV, por z^*(PA) o valor da solução ótima do PA com relaxação linear, e por z^*(PAD) o valor da solução ótima da relaxação linear dual do PA. Sabe-se que z^*(PA)=z^*(PAD). Entretanto, nota-se que z^*(PA)+$(c_{ij} - u_i - v_j)$ é um limite inferior sobre o custo de uma solução do PA que inclui o arco (i, j). [Miller & Pekny 91] fizeram uso deste algoritmo que retira inicialmente de consideração todas as variáveis x_{ij} cujos custos c_{ij} excedem o valor limite λ. Considerando um problema modificado PCV' com uma relaxação linear associada PA' e seu dual PAD', obtidos pela redefinição dos custos c_{ij} como segue:

$$c'_{ij} = \begin{cases} c_{ij}, & \text{Se } c_{ij} \leq \lambda \\ \infty, & \text{Caso Contrário.} \end{cases} \tag{III.30}$$

Os autores provaram a seguinte proposição a qual eles usaram como base para seu algoritmo:

Uma solução para o PCV' é ótima para o PCV se

$$z^*(PCV') - z^*(PA) \leq \lambda + 1 - u'_i - v'_{max} \tag{III.31}$$

e

$$\lambda + 1 - u'_i - v'_{max} \geq 0 \tag{III.32}$$

para $i = 1, \ldots, n$, onde u' e v' são soluções ótimas do PAD', e v'_{max} subestima o menor custo reduzido de qualquer variável descartada.

O algoritmo geral pode ser colocado então como:

Passo 1: (Inicialização) Escolha λ.

Passo 2: (Solução do PCV') Construa (c'_{ij}) e resolva o PCV'.

Passo 3: (Teste de Finalização) Se III.31 e III.32 verificam, então z^*(PCV')=z^*(PCV), pare. Senão, dobre λ e vá ao passo 2.

Os autores colocam que λ pode ser escolhido de forma razoável no *passo 1*, através por exemplo do arco de maior custo, e com ele é raro a necessidade de se realizar uma nova iteração. Para resolver o PCV' eles desenvolveram uma técnica B&B especial baseada na relaxação do PA, e algumas considerações fortes

sobre tornar uma solução do PAD em um ciclo hamiltoniano através de técnicas construtivas de um grafo admissível que por definição sua formação depende dos valores de custo original do grafo em análise e das variáveis duais do problema, de técnicas de *patching*, e usando o algoritmo de *matching bipartido não valorado* para saber o quão próximo um grafo admissível está de um verdadeiro circuito hamiltoniano.

A grande importância desse trabalho recai em que problemas assiméticos de tamanho razoável de 1.000 a 5.000 vértices foram resolvidos em torno de 4 a 38s usando estações Sun 4/330. Instâncias maiores de problemas assimétricos gerados randomicamente com 30.000 até 500.000 vértices foram resolvidas em computadores Cray YMP e Cray 2 entre 1.200s e 12.623s, respectivamente.

O mesmo algoritmo também foi testado para instâncias Euclidianas correspondente a pontos randomicamente colocados em um quadrado com lado de comprimento unitário. Os limites inferiores do PA mostraram-se, como se esperava, muito fracos fazendo com que o algoritmo resolvesse apenas pequenos problemas na otimalidade. Pelo que foi mostrado pelos autores, desaconselha-se o uso deste método nos casos simétricos e euclidianos.

III.3.2 Algoritmos Heurísticos Padrões

Devido às dificuldades relacionadas à resolução exata de algoritmos do PCV, muito se tem feito no desenvolvimento de heurísticas que produzem soluções próximas do ótimo em tempo computacional razoável. As heurísticas do PCV podem ser classificadas como:

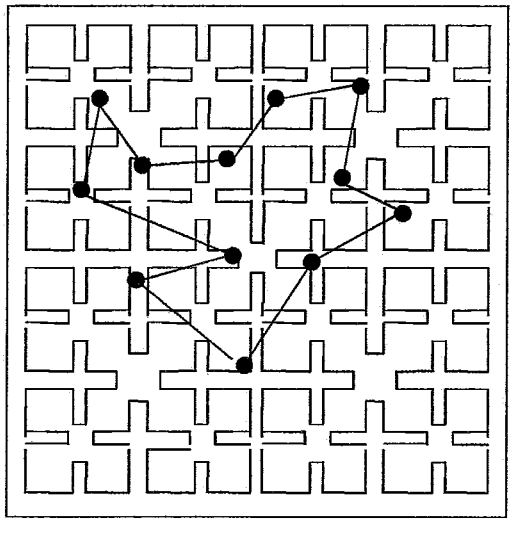
Algoritmos de Construção de Rotas: Constroem um ciclo hamiltoniano a partir da matriz de distâncias original;

Algoritmos de Melhoria de Rotas: Pegam um ciclo hamiltoniano e tentam melhorá-lo realizando pequenas modificações e

Algoritmos de Composição de Rotas: Combinam os dois acima aplicando uma melhoria a partir de um ciclo construido.

Nas aplicações com um grande número de vértices, como em Cristalografia, heurísticas rápidas tais como a heurística de *Curvas de Preenchimento de Espaço* de [Platzman & Bartholdi 89] foram preparadas para este fim. Uma curva de preenchimento ϕ é uma linha fractal que preenche um plano e pode ser definida como um mapeamento contínuo para um intervalo unitário $I = [0, 1]$ sobre um quadrado de área unitária $A = [0, 1]^2$. A curva de preenchimento foi projetada de

forma a preservar a proximidade, pontos próximos no mapa I sobre pontos próximos no mapa A. Na resolução de um PCV com heurísticas de curvas de preenchimento calcula-se para cada vértice v um escalar θ tal que $v = \phi(\theta)$. Os vértices são então ordenados de acordo com os seus valores correspondentes a θ, e a ordem resultante dá a rota do PCV. A complexidade da heurística é $O(n \ log \ n)$. Porém a qualidade das soluções geradas é apenas razoável, ver figura III.4.



Solução de um PCV.

Figura III.4: Uma solução heurística com curva de preenchimento.

Quanto à distância da solução ótima, algumas heurísticas garantem que seus resultados atingem no pior caso um certo percentual da solução ótima, já outras estão na classe dos resultados empíricos.

Heurísticas que Garantem Desempenho de Pior Caso

Considerando um PCV simétrico definido sobre um grafo G e onde C satisfaz à desigualdade triangular. Seja z^* o valor da solução ótima do PCV. Um modo simples de conseguir um limite inferior de z^* é encontrar o comprimento da árvore geradora mínima T sobre G. Como pode ser visto em [Weiss 92], a solução de T pode ser encontrada em $O(n^2)$. Chamando de $l(T)$ o comprimento da árvore. Uma estratégia possível é atravessar a árvore geradora ao longo de seus elos da seguinte forma:

Passo 1: Considere qualquer folha i_0 (vértice de grau 1) da árvore geradora e faça $i := i_0$.

Passo 2:

- Se há qualquer elo não percorrido (i, j) incidente ao vértice i, seguir o elo até o vértice j. Faça $i := j$ e repita o passo 2;
- Se todos os elos do vértice i foram atravessados, volte ao vértice k de onde i foi inicialmente atingido. Se $k := i_0$, pare; caso contrário faça $k := i$ e repita o passo 2.

Ilustrando o procedimento acima, considera-se a árvore geradora mínima na figura III.5. Iniciando com $i_0 = 1$, pode-se encaminhar para 3, 2, 3, 4, 5, 6, 5, 8, 5, 7, 5, 4, 3 e 1. É fácil ver que cada elo da árvore é coberto exatamente duas vezes. Em geral esta solução não é um ciclo hamiltoniano. Caso se deseje obter um ciclo hamiltoniano, o procedimento acima deve ser modificado com o uso de *cortes curtos*. Seguir o caminho obtido, pulando qualquer vértice já visitado. No exemplo, poder-se-ia seguir pelo caminho (1, 3, 2, 4, 5, 6, 8, 7, 1). Uma vez que está satisfeita a desigualdade triangular, nunca aumentando a distância total do ciclo e além disso, o caminho fechado resultante é uma rota de comprimento que não excede $2l(T) \leq 2z^*$.

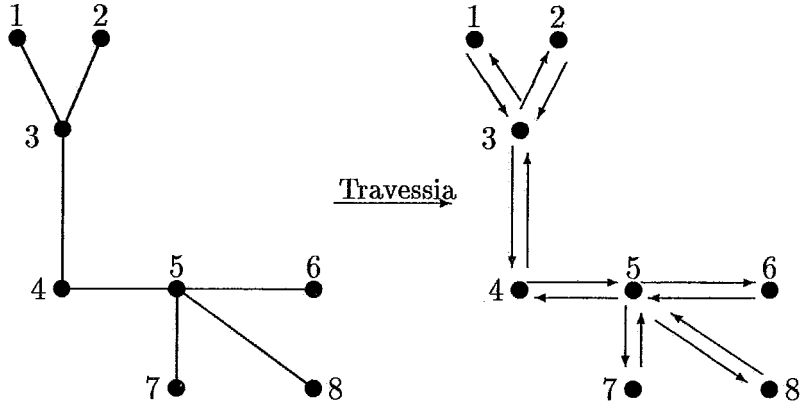


Figura III.5: Travessia ótima de um grafo usando Árvore Geradora Mínima.

Em [Christofides 76] tem-se uma proposta de melhoria do procedimento acima. Ela está baseada na seguinte observação. A árvore geradora mínima não é euleriana, entretanto, um grafo euleriano pode ser obtido dela pela união de vértices de grau ímpar através do algoritmo de 1-*matching* de custo mínimo. É possível obter este resultado em $O(n^3)$, [Papadimitriou & Steiglitz 82]. Novos elos correspondentes à solução ótima do problema de 1-*matching* são então incluídos na árvore. Seja $l(M)$ o comprimento total dos elos incluidos, o grafo Euleriano resultante e sua travessia completa requer uma distância total de $l(T) + l(M)$. Usando

novamente *cortes curtos*, um ciclo pode ser obtido sem que seja excedido o comprimento total de $l(T) + l(M)$, ver figura III.6 Nota-se que na figura III.6 o segundo grafo é obtido pela resolução do problema de *matching* de custo mínimo; os elos adicionados são mostrados através dos elos duplos. A travessia do grafo resultante pode ser feita pelo caminho (1, 2, 3, 4, 5, 6, 8, 5, 7, 5, 3, 1). Usando *cortes curtos* chegamos à rota (1, 2, 3, 4, 5, 6, 8, 7, 1).

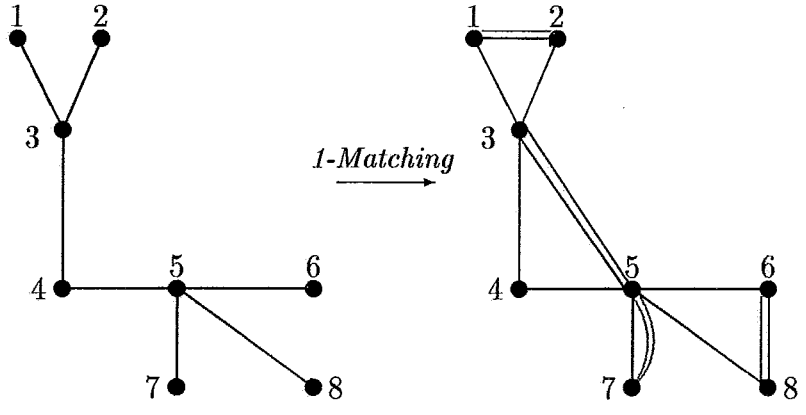


Figura III.6: Travessia ótima de um grafo usando Árvore Geradora Mínima e o Matching de Custo Mínimo.

Focalizando agora o custo adicional $l(M)$ e considerando que a sequência de vértices na solução ótima do PCV e a ligação por *cortes curtos*, tem-se todos os pares de vértices correspondendo aos elos na solução ótima do problema de *matching*, que retira todos os vértices intermediários. O comprimento total destes elos é $l(M)$. Agora fechando a rota pela ligação destes elos na mesma ordem que eles aparecem no ciclo. Os novos elos introduzidos têm um comprimento total de $l(M') \leq l(M)$ uma vez que o primeiro *matching* é ótimo. Como $l(M) + l(M') \leq z^*$, temos que $l(M) =\leq \frac{z^*}{2}$ e portanto a heurística permite uma rota de comprimento que não excede $l(T) + l(M) \leq \frac{3z^*}{2}$.

As diferenças básicas deste método no caso assimétrico está na geração da árvore geradora mínima onde se calcula a árvore geradora direcionada mínima e no caso do problema de matching, aplica-se o problema de transporte de modo a transformar o grafo obtido em euleriano, [Lawler et al 85].

Heurísticas com Bom Desempenho Empírico

Nos procedimentos heurísticos de construção de rotas do Caixeiro Viajante, tem-se os métodos de formação de rotas entre vértices próximos/distantes,

métodos de inserção, métodos de *patching*, [Rosenkrants et al 77], [Or 76] e [Karp 76]. Os métodos gerais são discutidos a seguir.

1. *Algoritmos do Vizinho* mais Próximo/Mais Distante [1] - O(n^2), Simétrico/Assimétrico

   Passo 1: Definir o vértice inicial;

   Passo 2: Determinar o vértice mais próximo/distante do último vértice incluido na rota do caixeiro. Se há vértice ainda não considerado repetir este passo.

   Passo 3: Unir o último vértice da rota ao primeiro.

2. *Algoritmos de Inserção* - O(n *log* n) até O(n^2)

   Passo 1: Construir uma rota consistindo de dois vértices, incluido o vértice origem.

   Passo 2: Considerar todos os vértices que ainda não estão na rota. Inserir na rota um vértice escolhido com respeito a uma dada estratégia, por exemplo:

   - O vértice que permite a menor distância de incremento (Simétrico/Assimétrico);
   - O vértice mais próximo à rota corrente (Simétrico/Assimétrico);
   - O vértice que forma o maior ângulo entre dois vértices consecutivos na rota, (Simétrico) etc.

3. *Algoritmos de* **Patching** (Remendos) - O($n^2 + m$) - Assimétrico

   Este procedimento explora o fato de que sendo c_{ij} uniformemente distribuido, a relaxação de atribuição do PCV atinge soluções próximas ao ótimo, [Lawler et al 85].

   Passo 1: Resolver o problema de Atribuição para a matriz de custos C associada ao PCV.

   Passo 2: Se a solução contém apenas um circuito, parar.

   Passo 3: Selecionar dois circuitos contendo o maior número de vértices. Selecionar um arco $< i, j >$ e um arco $< k, l >$ no segundo circuito que minimiza o custo de junção dos circuitos, $c_{il} + c_{kl} - c_{ij} - c_{kl}$. Ir para o passo 2.

4. *Algoritmos de Melhoria*

   (a) **Algoritmos k-Otimais**[2]

[1] Conjunto de métodos implementados no **SisGRAFO**
[2] Conjunto de métodos implementados no **SisGRAFO**

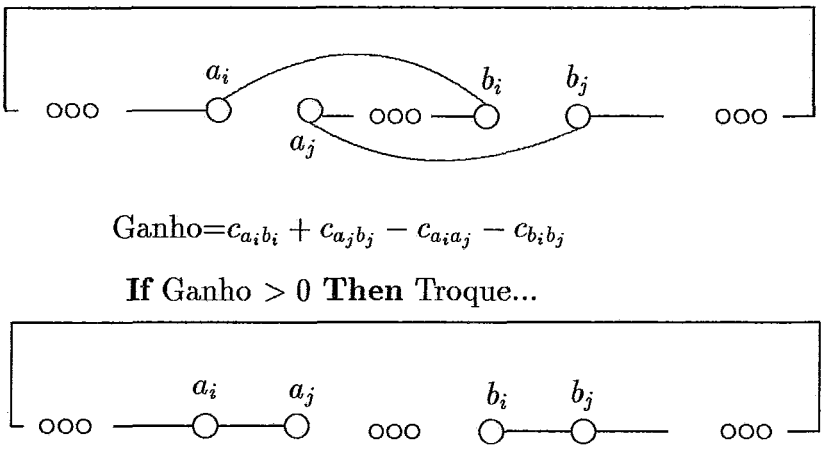


Figura III.7: Trocas 2-Otimais no caso simétrico.

Passo 1: Considerar uma rota inicial;

Passo 2: Remover k arcos da rota e reconectar por tentativa as k cadeias restantes de todas as formas possíveis. Se qualquer reconecção permite uma rota menor, considerar esta rota como uma nova solução inicial e repetir o passo 2. Parar quando nenhuma melhora puder ser obtida.

No caso considerado nota-se que a cada passo há n^k modos de remover arcos e $k!$ de reconectar as cadeias não direcionadas resultantes. Nenhuma destas reconecções são, entretanto, viáveis. [Croes 54] e [Lin 65] utilizaram k como 2 e 3, ver figura III.7. [Lin & Kerningham 73] propuseram um método de melhorias a este algoritmo básico, onde a idéia principal é encontrar o valor de k apropriado, sendo o mesmo modificado dinamicamente através do algoritmo. Este procedimento é bem mais difícil de se implementar computacionalmente do que o método original k-otimal de Lin, além disto ele geralmente produz soluções muito próximas ao ótimo. [Johnson 90] desenvolveu um método de Lin-Kerningham interativo randômico cujos resultados atingem valores 2% além da solução ótima, já [Mak & Morton 93] buscaram ampliar o espaço de busca em cada passo da pesquisa em profundidade do método de Lin-Kerninghan e com poucas linhas de código conseguiram atingir resultados expressivos em tempo computacional e qualidade nas soluções. O grande mérito deste último trabalho está na possibilidade de se manter a qualidade dos resultados de Lin-Kerningham sem que para isto seja necessário um esforço computacional extra nas buscas, além disso mantém de certa forma os resultados em torno de 2% a 4% acima da solução ótima.

Na mesma linha dos métodos otimais [Or 76] propôs um método de trocas bastante simplificado de ordem $O(n^2)$ em cada passo, porém produzindo rotas tão boas quanto aquelas encontradas pelo algoritmo 3-otimal. O algoritmo de Or pode ser descrito como segue.

(b) **Algoritmo de OR**

Passo 1: Considerar uma rota inicial e fazer $t := 1$ e $s := 3$ [3];
Passo 2: Remover da rota uma cadeia de s vértices consecutivos, iniciando com um vértice na posição t, e por tentativa inseri-lo entre todos os pares restantes de vértices consecutivos sobre a rota:

- Se a tentativa de inserção decresce o custo da rota, implementá-la imediatamente, definir a nova rota; fazer $t := 1$ e repetir o passo 2.
- Se nenhuma tentativa de inserção decresce o custo da rota, fazer $t := t + 1$. Se $t := n + 1$, então proceder o passo 3, caso contrário repetir o passo 2.

Passo 3: Faça $t := 1$ e $s := s - 1$. Se $s > 0$ ir para o passo 2, caso contrário parar.

As técnicas acima apresentadas são discutidas com bastante ênfase em [Nurmi 91], que implementou e discutiu a maioria das técnicas acima discutidas, inclusive a modificação do algoritmo otimal de [Kanellakis & Papadimitriou 80], bastante superior às técnicas otimais para o caso assimétrico, III.8. Neste trabalho mostrou-se que empiricamente os resultados obtidos com estas heurísticas para o caso assimétrico não atingiram soluções próximas do ótimo. Este fato pudemos comprovar em nossas avaliações no **SisGRAFO**. Uma idéia geral sobre o funcionamento da heurística 2-Otimal para o caso assimétrico é apresentada na figura III.9.

(c) *Algoritmos Baseados em Metaheurísticas*

Metaheurísticas são um conjunto de regras universais de um determinado contexto as quais podem se adaptar às estruturas com arquiteturas paralelas e são direcionadas à busca de um ótimo global de um determinado problema, [Ochi 94].

Em muitos problemas combinatórios, o uso destas técnicas vem obtendo excelentes resultados em desempenho computacional e na qualidade das soluções, principalmente no que concerne a problemas NP-Árduos como o PCV. As principais Metaheurísticas que surgem neste contexto são *Simulated Annealing, Algoritmos Genéticos e Busca Tabu*, as quais são discutidas superficialmente a seguir.

[3] [Nurmi 91] avaliou que se $s := 5$ os resultados computacionais serão sempre melhores do que caso se inicie de $s := 3$.

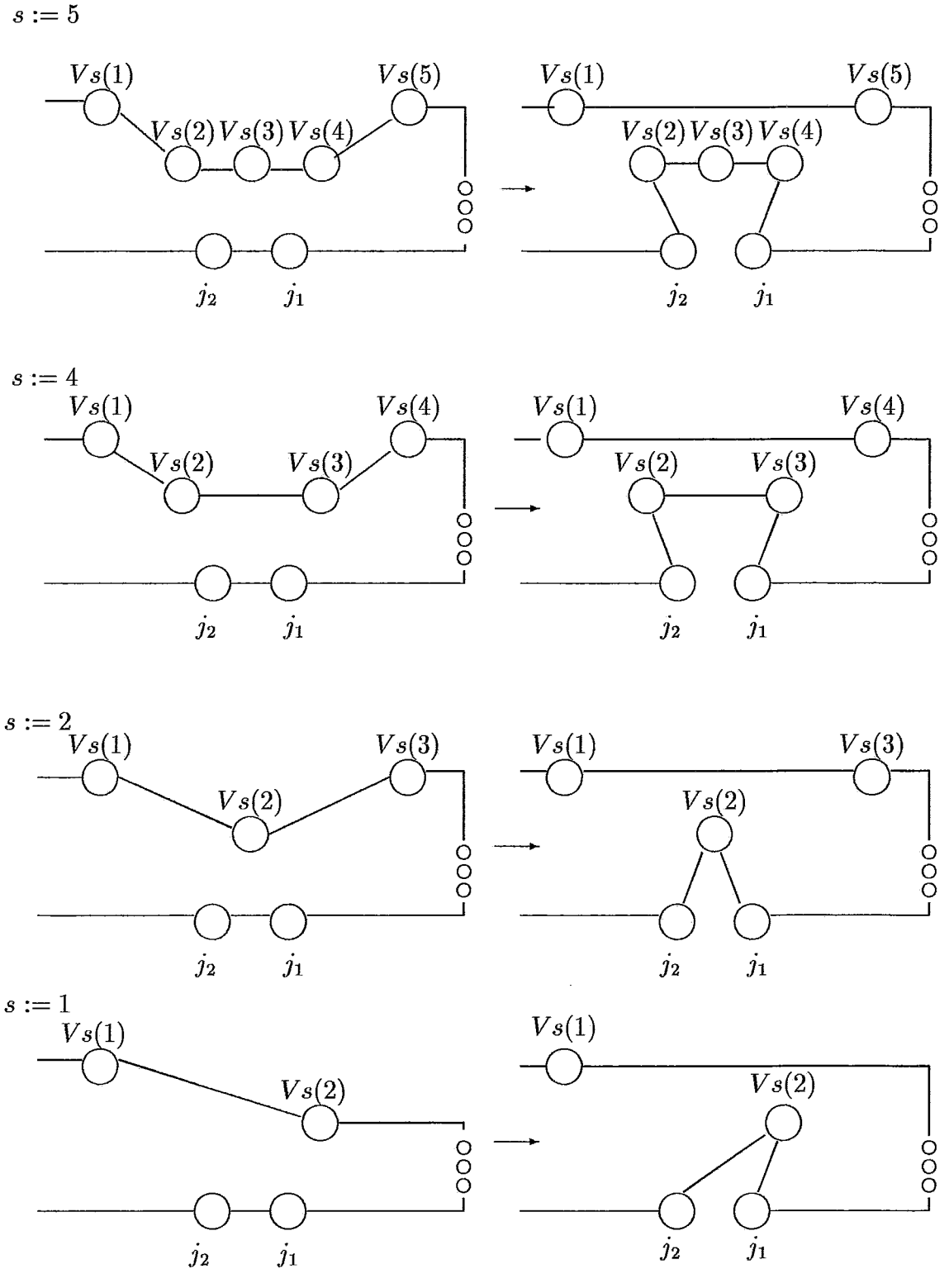
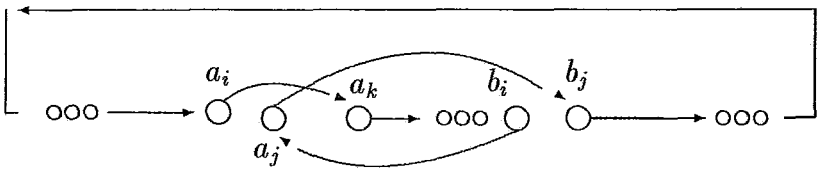


Figura III.8: Trocas Or-Otimais no caso simétrico.



Ganho=$c_{a_ia_k} + c_{a_jb_j} + c_{b_ia_j} - c_{a_ia_j} - c_{a_ja_k} - c_{b_ib_j}$

If Ganho > 0 **Then** Troque...

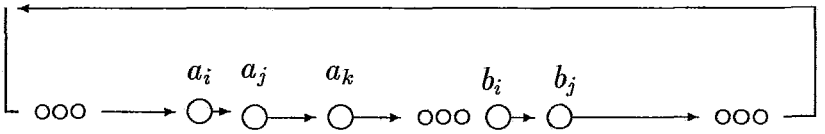


Figura III.9: Trocas 2-Otimais no caso Assimétrico.

i. *Simulated Annealing*
É um método de melhorias sucessivas que deriva do processo de *annealing* (recozimento, têmpera) de materiais usados na mecânica. Com o objetivo de fazer com que um material atinja o seu estado sólido de mínima energia, é necessário aquecê-lo até que suas partículas sejam randomicamente distribuidas no estado líquido; então, para evitar mínimos locais, sua temperatura é reduzida gradualmente passo a passo, até que o sistema adquira o estado de equilíbrio para um determinado nível de temperatura T, todos os estados possíveis podem ser atingidos mas enquanto o sistema esfria, o número de possibilidades é reduzido e o processo converge a um estado estável.
Nos problemas de combinatória, o objetivo é sair de uma determinada solução inicial para uma solução de custo mínimo, pela execução de mudanças graduais na solução inicial. Chamando de T o estado do processo (T corresponde ao nível de temperatura de um sistema físico). No início, o valor de T é alto e o número de movimentos permitidos é também alto. Este número decresce com T, até que nenhuma mudança na solução seja possível. O mínimo local foi alcançado.
Para um dado valor de T, o algoritmo é parecido com o algoritmo k-otimal, todas as soluções y em uma *vizinhança* de uma solução x são avaliadas. Entretanto, substituir x por y é algumas vezes permitido mesmo que o resultado leve a um valor mais alto. Isto reduz a probabilidade de se cair numa armadilha de um ótimo local. De um modo mais formal um algoritmo geral para aplicação de *Simulated Annealing* pode ser resumido como:

Passo 1: Considerar uma rota inicial x de custo $F(x)$.

Passo 2: Considerar uma solução y de custo $F(y)$ na vizinhança de x.

Se $F(y) < F(x)$, fazer $x := y$ e repetir o passo 2.

Se $F(y) \geq F(x)$, definir $p_T e^{\frac{[F(x)-F(y)]}{T}}$, onde T é um parâmetro tendendo a zero à medida que o processo evolui.

Selecionar aleatoriamente um número k em [0,1]. Se $k \leq p_T$, fazer $x := y$ e repetir o passo 2. Senão, repetir o passo 2 com uma nova solução y na vizinhança de x, ou parar se esta vizinhança já foi completamente avaliada.

ii. *Algoritmos Genéticos*

São técnicas que surgiram baseadas nos princípios da evolução natural e hereditariedade, enquadrando-se a Programação Evolutiva onde os Algoritmos Genéticos estão inseridos, [Ochi 94]. A Programação Evolutiva é um Algoritmo Genético probabilístico iterativo que mantém a cada iteração t uma população de indivíduos (cromossomos) $P(t) = (p_1^t, p_2^t, \ldots, p_n^t)$.

O processo de renovação iterativa das populações pode ser descrito formalmente do seguinte modo: cada indivíduo p_i^t para todo $i = 1, \ldots, n$ representa a solução de um problema associado. Um mecanismo permite a renovação da população $P(t)$ a partir da população $P(t-1)$. Estes mecanismos são chamados de Operadores Genéticos, onde se define regras para uma renovação eficiente da população. A cada passo os indivíduos são avaliados segundo uma função que mede o nível da sua aptidão, por critérios que dependem do problema em análise. Assim, os indivíduos *mais aptos* sobrevivem para a próxima iteração e os menos aptos são descartados.

A notação dos algoritmos genéticos pode ser relacionada à notação matemática de conjuntos soluções conforme a tabela III.1.

Algoritmos Genéticos	Matemática
Uma População de Cromossomos	Um conjunto de soluções
Um Cromossomo	Uma solução
Um Gene	Um componente de uma solução

Tabela III.1: Representação de Algoritmos Genéticos em Matemática.

O Algoritmo Genético padrão é descrito como:

Passo 1: Fazer $t := 0$. Gerar a população $P(t)$.

Passo 2: **Avaliar** $P(t)$. Se $P(t)$ satisfaz às condições de parada, Terminar.

Passo 3: Fazer $t := t + 1$. **Selecionar** $P(t)$ de $P(t-1)$. Aplicar uma **Recombinação de Operadores** de $P(t)$. Ir ao Passo 2.

Os procedimentos **Avaliar**, **Selecionar** e **Recombinar Operadores** são estratégias dependentes do tipo de problema a ser resolvido.

iii. *Algoritmos de Busca Tabu*

Assim como nos métodos anteriores, vizinhos sucessivos de uma solução x são verificados e, assim como nos algoritmos de SA e GA permite-se piorar o valor da função objetivo de modo a evitar o ótimo local. A fim de evitar ciclar numa solução, as soluções que já foram avaliadas são proibidas e inseridas em uma lista que fica constantemente sendo atualizada chamada de *Lista Tabu*, [Glover 89] e [Glover 90a]. O algoritmo geral de uma Busca Tabu pode ser sumarizado como:

Passo 1: Considerar uma solução inicial x de custo $F(x)$. Fazer a Lista Tabu $T = \emptyset$.

Passo 2: Seja $N(x)$ a vizinhança de x. Se $N(x) \setminus T = \emptyset$, ir para o passo 3. Caso contrário, identificar uma solução de menor custo y em $N(x) \setminus T = \emptyset$ e fazer $x := y$. Atualizar T e a melhor solução encontrada.

Passo 3: Se o número máximo de iterações permitidas desde o início do processo até a última atualização foi atingido, terminar. Caso contrário ir para o passo 2.

Este método, assim como os anteriores desta classe, dependem muito da escolha de parâmetros de controle.

5. *Algoritmos de Composição*

Dois algoritmos se destacam basicamente nesta classe. O primeiro é a heurística chamada de CCAO, [Golden & Stewart 85] e a segunda chamada de GENIUS, [Gendreau et al 90].

A heurística CCAO foi projetada para as instâncias euclidianas do PCV. Ela explora as propriedades destes problemas, mostrando que em qualquer solução ótima deste caso, os vértices localizados na envoltória convexa de todos os vértices, são visitados na ordem na qual eles aparecem nos limites da envoltória. O método pode ser resumido como segue:

O Algoritmo CCAO

Passo 1: (Envoltória Convexa - C: *Convex Hull*) Definir uma rota parcial inicial formada pela envoltória convexa dos vértices.

Passo 2: (Inserção Mais Barata - C: *Cheapest Insertion*) Para cada vértice k não inserido na rota, identificar os dois vértices adjacentes i_k e j_k sobre a rota de modo que $c_{i_k k} + c_{k j_k} - c_{i_k j_k}$ é minimizado.

Passo 3: (O Maior Ângulo - A: *Largest Angle*) Selecionar o vértice k^* que minimiza o ângulo entre os elos (i_k, k) e (k, j_k) sobre a rota, e inserí-los entre i_k^* e j_k^*.

Passo 4: Repetir os passos 2 e 3 até que um ciclo hamiltoniano passando por todos os vértices seja obtido.

Passo 5: (O: Or-Otimal) Aplicar o procedimento Or-otimal na rota. Terminar.

A grande vantagem deste método está justamente nos passos 2 e 3 onde a heurística através da seleção de k^* busca maximizar o ângulo que ele faz com a rota, mantendo a solução sempre o mais próximo possível da envoltória convexa.

O Algoritmo GENIUS

A maior desvantagem, porém, do algoritmo CCAO é a fase de inserção que é míope quando se considera que as inserções são executadas sequencialmente sem muita preocupação com o ótimo global. As inserções podem em algum momento atingir decisões muito ruins de tal modo que a fase de pós-otimização será incapaz de identificar o problema. A heurística GENIUS executa cada inserção de uma forma mais cuidadosa, através da realização de um número limitado de transformações locais da rota, e simultaneamente com a inserção dela mesma. Ela consiste de duas partes: uma fase de inserção generalizada, seguida de uma fase de pós-otimização que remove sucessivamente os vértices da rota e os reinsere, usando uma regra generalizada de inserção.

Estes dois últimos métodos mostraram-se comparáveis aos métodos Metaheurísticos na maioria dos problemas testes da literatura.

Capítulo IV

Generalizações do Problema do Caixeiro Viajante

Tomando a definição formal do PCV, seja $G(V, L)$ um grafo completo com um conjunto de vértices V e um conjunto de ligações L, estende-se a aplicação do PCV com restrições supondo que existem um conjunto P de atributos nos vértices que correspondem a valores, frequências ou custos, e uma matriz C de custos associados às ligações. Deseja-se então encontrar um subconjunto V' tal que $V' \subseteq V$, de modo que V' forme um ciclo hamiltoniano de custo mínimo considerando os custos de P e C sujeito ao conjunto de restrições R sobre P e/ou C. Esta classe de problemas mantém a complexidade do PCV, uma vez que um dos subproblemas é o próprio PCV.

Um grande número de aplicações surgem a partir desta generalização do PCV, a seguir temos algumas das quais vêm sendo exploradas com ênfase pela literatura.

1. **Problema do Viajante de Compras** - (*Traveling Purchaser Problem*)
   Onde dado que um viajante parte de um local de origem, chamado de s, um conjunto $I = \{1, 2, \ldots, m\}$ de lojas, e um conjunto de $K = \{1, 2, \ldots, k\}$ de produtos, o problema do Viajante de Compras (PVC) é gerar um ciclo através de um subconjunto de lojas de I e sua residência, e comprar cada um dos n itens específicos apenas uma vez nestas lojas de modo que o custo total de viagem e compra seja minimizado.

   Admite-se que cada produto está disponível em pelo menos um mercado, que o viajante pode passar por uma loja quantas vezes quiser sem comprar nada, que o viajante deve comprar todos os tipos de produtos disponíveis, e que não há produtos disponíveis em seu local de partida.

   Este problema se aplica na manufatura flexível, na análise de testes bio-

químicos, no roteamento e sequenciamento de veículos, dentre outras. Formulações matemáticas, e heurísticas para o problema são encontradas em [Golden et al 81], [Ochi & Maculan 91] e [Maculan et al 92].

2. **O Problema de Compras** - (*Shopping Problem*)

   É uma extensão do problema do viajante de compras, porém considerando a hipótese de se adquirir mais de uma unidade de tipo de produto por loja a partir de uma lista de necessidades. Uma loja pode não ter em estoque a quantidade total requerida de um certo produto, permitindo com isso que se compre o mesmo produto em lojas distintas. O PC pode ser aplicado na manufatura flexível, no sequenciamento de tarefas e no roteamento de veículos vendedores de massas e biscoitos, [Negreiros et al 94].

3. **O Problema do Caixeiro Viajante Restrito no Tempo** - (*Time-Constrained Traveling Salesman Problem*)

   Supondo que cada par de pontos (i, j) esteja associado a um lucro e a um tempo de viagem. O objetivo do PCV Restrito no Tempo (PCVRT), é encontrar a subrota que inicia e termina num ponto origem, que maximiza o lucro enquanto está restrito a τ unidades de tempo para a tarefa. Este problema tem aplicação na indústria, e no dimensionamento de territórios de vendas. Para ele um algoritmo heurístico pode ser encontrado em [Golden et al 81].

4. **O Problema do Caixeiro Viajante Coletor de Valores** - (*Prize Collecting Traveling Salesman Problem*)

   Dado um depósito de onde parte um veículo, e um conjunto de cidades, cada uma delas tendo um preço associado p_i, seja C a matriz de custos e γ_i o custo de deixar a cidade i sem que seja visitada. O Problema do Viajante Coletor de Valores (PVCV) é encontrar a rota de mínimo custo do veículo, que visita as cidades pelo menos uma vez coletando a um custo nunca inferior a g.

   Este problema surgiu como um modelo para o sequenciamento da operação diária de uma fábrica de rolamentos de aço. Uma fábrica de aço produz chapas grossas através de rolamentos frios ou quentes. Seja c_{ij} o custo de proceder o pedido j logo após o pedido i, e p_i o peso da chapa grossa atribuida ao pedido i. O planejamento da operação diária consiste em selecionar um subconjunto de pedidos que satisfaçam a um determinado limite inferior g sobre o peso total, e sequenciá-los de modo a minimizar o custo total, [Balas & Martin 85], [Balas 89] e [Fichetti & Toth 88].

   Um algoritmo B&B para o problema foi proposto por [Fichetti & Toth 88], que reporta instâncias com até 200 vértices para grafos simétricos, euclidianos e assimétricos, variando as instâncias a partir de um multiplicador dos pesos, com o objetivo de reduzir a sua importância na formação dos circuitos de coleta.

5. **O Problema de Orientação** - (*Orientering Tour Problem* - PRO)

   Orientação é um esporte onde cada participante seleciona um caminho de um início a um destino pré-definidos, com o objetivo de visitar pontos de controle que se encontram ao longo do caminho. Cada ponto de controle tem uma pontuação associada, e os percursos entre os pontos de controle têm seus custos. O problema é de selecionar um conjunto de pontos de controle a visitar, tal que o escore seja maximizado sujeito a uma restrição de movimentação (tempo máximo) sobre o custo total.

   Como se pode ver este problema tem algumas caracteristicas do PVCV, porém com a liberdade de término em um local diferente daquele de partida além da ausência de penalidade por se deixar de visitar um determinado ponto. Uma outra denominação para este problema foi dada por [Laporte & Martelo 90] como o *Problema do Caixeiro Viajante Seletivo.*

   O PRO pode ser encontrado em um número grande de aplicações, como no roteamento de tanques de óleo que atendem a navios em vários locais, na atribuição de veículos a clientes e em problemas de roteamento e estocagem, [Golden et al 84a] e [Golden et al 87]. Pode também ser visto como uma extensão do PVCV, tendo sido bastante explorado pela literatura através de métodos exatos baseados em relaxação lagrangeana e B&B, e heurísticos, [Golden et al 88], [Ramesh 91], [Laporte & Martelo 90] e [Ramesh et al 92].

6. **O Problema do Caixeiro Viajante Multiperiódico** - *Multiperiod Traveling Salesman Problem*)

   Este problema considera que um conjunto de cidades serão visitadas numa frequência conhecida sobre um horizonte tempo de M-dias.

   Colocando mais formalmente o Problema do Viajante Multiperiódico (PVMP). Seja uma cidade base rotulada como 0, e um conjunto (V) n de cidades ($I = \{1, \ldots, n\}$) a serem visitadas, uma matriz de custos C entre os pares de cidades $(i, j) \in I \cup \{0\}$, um horizonte de tempo de M-dias $H = \{1, \ldots, M\}$ e uma política fixa de periodicidade sendo dada.

   A política de visitas impõe as seguintes restrições: Através do horizonte H, cada cidade $i \in I$ deve ser visitada somente uma vez e cada intervalo de tempo de T_i dias ($1 \leq T_i \leq M$) sobre as restrições adicionais de que em pelo menos K_i dias ($1 \leq K_i \leq T_i$) e em pelo menos U_i dias ($T_i \leq U_i \leq 2T_{i-1}$) deve-se passar em i entre dois dias de visitas sucessivos, respectivamente.

   Consequentemente, a k-ésima visita da cidade $i \in I$ deve ocorrer no dia v^k do conjunto $S^k = \{(k-1)T_i + 1, (k-1)T_i + 2, \ldots, kT_i\}$ representando os dias do k-ésimo intervalo de tempo para a cidade $i \in I$. Também, duas sucessivas ocorrências de visita dos dias para a cidade $i \in I$, $v^k \in S_i^k$ e $v^{k+1} \in S_i^{k+1}$ devem satisfazer à restrição $K_i \leq v^{k+1} - v^k \leq U_i$. Em geral, M é considerado como o menor múltiplo comum dos T_i's de modo que se

possa levar em conta todos as combinações possíveis dos dias de visita. Portanto, a taxa $R_i = \frac{M}{T_i}$ é um inteiro que representa o número de visitas que a cidade i receberá do viajante no horizonte de tempo M.

Qualquer conjunto de R_i dias que satisfaça às restrições periódicas estão referenciadas como uma combinação de dias de visita para a cidade $i \in I$, enquanto que um conjunto de n combinações de dias de visita para cada cidade $i \in I$, considerar-se-á como uma solução do dia de visita.

Assim, o problema é de atribuir uma combinação dos dias de visita à cidade $i \in I$ tal que a soma dos comprimentos das menores rotas (cada uma delas conectando as cidades que devem ser visitadas durante o mesmo dia à cidade base 0) seja minimizado.

O PCVMP pode ser modelado como um grafo se M grafos completos são associados a cada solução do dia de visita. Aqui, cada um dos M grafos é definido sobre um conjunto de cidades onde a solução do dia de visita impõe a visita durante o mesmo horizonte de tempo mais a cidade base, e o peso de cada ligação (a, b) é igual à distância entre as cidades.

O objetivo do PCVMP será, pois, de encontrar uma solução do dia de visita tal que o comprimento total dos dias de visita dos menores ciclos hamiltonianos sobre M grafos relativos ao dia de visita solução selecionado seja mínimo.

O PCVMP pode surgir em um número grande de aplicações, tais como no roteamento periódico de veículos, no escalonamento de força de vendas específicas com frequência de atendimentos, no escalonamento de trabalho de serviços hospitalares a enfermos, no escalonamento de turmas de varrição de ruas, dentre outros, [Christofides & Beasley 84], [Tan & Beasley 84] e [Galdioso & Palleta 92].

7. **O Problema do Caixeiro Viajante com Gargalos** - *Mini-Max Traveling Salesman Problem*)

   O objetivo do PCVG é determinar um percurso Hamiltoniano tal que a maior ligação entre duas cidades seja a menor possível. Este problema se aplica na Manufatura Flexível, onde a fabricação de um produto deve passar por n tarefas as quais serão realizadas num processo de montagem que deverá percorrer n estações de trabalho.

A seguir discute-se principalmente os problemas de Compras considerando sua formulação matemática e a proposta de um algoritmo heurístico. Logo em seguida extraiu-se uma abordagem do Problema de Orientação também considerando as formulações matemáticas e heurísticas.

IV.1 O Problema de Compras

Como foi visto acima, os modelos clássicos de Compras ou de Coletas, têm sido amplamente estudados na literatura afim, [Bodin et al 83], [Assad & Golden 88], mas com grande enfoque a problemas clássicos de distribuição e estoque. Na área de compras, os modelos estudados são para o caso com um único tipo de produto e/ou coleta de produto, sem custos diferenciados de aquisição. Já o Problema de Compras (*Shopping Problem*), considerando quantidades e custos diferenciados de aquisição, tem sido pouco explorado pela literatura, [Ramesh 78], [Golden et al 81], [Ochi & Maculan 91] e [Maculan et al 92].

O Problema de Compras é basicamente descrito, como foi visto brevemente, por uma lista de m tipos de produtos ($k = 1, \ldots, m$), por quantidade de cada produto k a ser adquirido (Q_k), e por uma lista de n centros ($j = 1, \ldots, n$) de onde devem ser adquiridos as quantidades acima. Um produto k pode ter custos de aquisição diferente em centros distintos e cada produto k deve estar disponível em pelo menos um centro j.

A seguir coloca-se a modelagem do Problema de Compras e a fomulação de uma heurística dinâmica, ressaltando porém que o algoritmo aqui proposto pode ser adaptado a outras variações de tipos de problemas de coletas e/ou distribuição.

IV.1.1 Formulações do Problema de Compras

O **PC** pode ser descrito do seguinte modo: *Seja um conjunto de centros de consumo* **CC**={ $cc_0, cc_1, cc_2, \ldots, cc_n$ } *e um conjunto de p produtos que se deseja adquirir* **PAdq**={$pa_1, pa_2, \ldots, pa_p$ }, *nos centros de consumo* cc_j ($j = 1, \ldots, n$). Supondo-se que cada centro de consumo cc_j possui uma dupla (e_j^k , d_j^k) *para* ($j = 1, \ldots, n$ e $k = 1, \ldots, p$), onde e_j^k é a quantidade do produto k disponível em cc_j, e d_j^k é o custo unitário (\$) do produto k em cc_j. Assumindo-se conhecida a matriz de custos (\$) **C** entre os centros de consumo, **C**=$[c_{ij}]$ ($i, j = 0, \ldots, n$), e conhecendo as quantidades desejadas de aquisição por produto Q^k ($k = 1, \ldots, p$), que é igual ou inferior à soma total de disponibilidades nos centros de consumo por produto. Deseja-se construir um circuito a partir de cc_0 e a ele retornando, que passe por um subconjunto de **CC**, com custo mínimo total de compra e deslocamento.

Assim como no **PVC**, [Ochi & Maculan 91], considera-se que $c_{ii} = 0$ ($i = 0, \ldots, n$), e que a matriz de custos das distâncias é completa ($0 \leq c_{ij} \geq \overline{c}$, $\overline{c} << \infty$, $\forall i, j$).

Seja então as seguintes variáveis de decisão para o Problema de Com-

pras:

$$x_{ij} = \begin{cases} 1, & \text{Se arco (i,j) é percorrido;} \\ 0 & \text{Caso contrário} \end{cases}$$

y_j^k : Quantidade do produto k adquirido em cc_j;

z_{ij}^k : Fluxo de produto k sobre o arco (i,j);

K, é o conjunto formado pelos p tipos de produtos existentes $\{k_1, \ldots, k_p\}$;

S, é o conjunto formado pelas restrições que proibem a formação de rotas ilegais do Caixeiro Viajante;

$\mathcal{P}(\mathcal{S})$, é uma partição viável do conjunto S que inclui sempre a origem nos extremos.

Um modelo de programação matemática para o problema pode ser assim formulado:

$$\textbf{(PC-1)} \text{ Minimizar } \sum_{i \in N} \sum_{j \in N} c_{ij}\, x_{ij} + \sum_{j \in N-\{0\}} \sum_{k \in K} d_j^k\, y_j^k \tag{IV.1}$$

$$sujeito\ a:$$

$$y_j^k \leq e_j^k, \ \forall j \in N - \{0\}, \ \forall\, k \in K \tag{IV.2}$$

$$\sum_{j \in N-\{0\}} y_j^k = Q^k, \ \forall\, k \in K \tag{IV.3}$$

$$\sum_{j \in N} x_{ij} - \sum_{j \in N} x_{ji} = 0, \ \forall i \in N \tag{IV.4}$$

$$\text{Se } \sum_{i \in N} x_{ij} \geq 0 \Rightarrow \sum_{k \in K} y_j^k \geq 0, \ \forall\, j \in N \tag{IV.5}$$

$$z_{ij}^k \leq Q^k\, x_{ij}, \quad i,j \in N, \ k \in K \tag{IV.6}$$

$$\sum_{i \in N-\{0\}} z_{ij}^k + y_j^k - \sum_{i \in N-\{0\}} z_{ji}^k = 0, \ \forall j \in N, \ k \in K \tag{IV.7}$$

$$\sum_{i \in N} z_{i0} = Q^k, \ \forall k \in K \tag{IV.8}$$

$$X = (x_{ij}) \in \mathcal{P}(\mathcal{S}) \tag{IV.9}$$

$$x_{ij} \in \{0,1\}, \ z_{ij}^k \geq 0, \ y_j^k \geq 0, \ \forall\, i,j \in N, \ \forall\, k \in K \tag{IV.10}$$

A função objetivo deseja minimizar o custo de percurso e o custo de aquisição dos produtos, IV.1. As restrições IV.2 garantem que a quantidade

de produto comprada por loja não excederá à quantidade em oferta na loja. As restrições IV.3 garantem que a solicitação por quantidade de produto será atendida. As restrições IV.4 garantem a conservação de fluxo ao longo do percurso definido pelo viajante de compras. Nas restrições IV.5 garantem que uma vez que se realize um percurso entre dois centros de consumo, haverá uma compra de algum produto k no centro destino j. As restrições IV.6 consideram que a quantidade de fluxo por produto k que flui num arco $< i, j >$ é sempre limitada à quantidade demandada do produto k. As restrições IV.7 consideram que o fluxo de um produto k que flui passa por um vértice j a menos da origem, recebe uma quantidade y_j^k do produto ao sair do vértice. As restrições IV.8 garantem que o fluxo de produto k chegando na origem será sempre igual à demanda de compra do produto k. As restrições IV.9 são as restrições de eliminação de subrotas ilegais permitindo somente uma partição conexa do conjunto S que utiliza o vértice origem. Por fim as restrições IV.9 definem o espaço das variáveis do modelo.

Uma outra formulação mais compacta, sugerida por [Maculan 96], considera $G = (V, A)$ um grafo orientado e completo sem laços, onde $V = \{0, 1, \ldots, n\}$ e $A \subset V \times V$, onde o vértice 0 é o ponto de partida e destino final do veículo, e os demais correspondem a centros de vendas. Admitindo as mesmas condições de produtos e custos da formulação anterior (onde todo $c_{ij} > 0$), e supondo que a capacidade do veículo seja maior ou igual a $\sum_{k \in K} Q^k$, deseja-se encontrar um percurso para o veículo visando à compra da quantia total da demanda, minimizando o custo de compra adicionado ao custo de transporte.

As seguintes variáveis são necessárias ao modelo:

$$x_{ij}^k = \begin{cases} 1, & \text{A quantia do produto } k \text{ transportada no arco } (i,j) \in A; \\ 0 & \text{Caso contrário} \end{cases}$$

$u_j^k \geq 0$: Quantidade do produto k adquirido em cc_j, $j \in V - \{0\}$;

$$y_{ij} = \begin{cases} 1, & \text{Se o arco } (i,j) \text{ é usado na rota do veículo ;} \\ 0 & \text{Caso contrário} \end{cases}$$

K, é o conjunto formado pelos p tipos de produtos existentes $\{k_1, \ldots, k_p\}$;

$\Gamma^+(i)$, é o conjunto de sucessores de i;

$\Gamma^-(i)$, é o conjunto dos predecessores de i.

O modelo proposto é o que segue:

$$\textbf{(PC-2)} \text{ Minimizar } \sum_{(i,j) \in A} c_{ij}\, y_{ij} + \sum_{i \in V - \{0\}} \sum_{k \in K} d_j^k\, u_j^k \qquad \text{(IV.11)}$$

$$sujeito\ a:$$

$$\sum_{j\in\Gamma^-(i)} x_{ij}^k + u_i^k - \sum_{j\in\Gamma^+(i)} x_{ji}^k = 0, \quad \forall j \in V-\{0\},\ k \in K \tag{IV.12}$$

$$\sum_{i\in\Gamma^-(0)} x_{i0}^k = Q^k, \quad \forall k \in K \tag{IV.13}$$

$$0 \leq u_i^k \leq q_i^k,\ i \in V-\{0\}, k \in K \tag{IV.14}$$

$$\sum_{j\in\Gamma^+(i)} y_{ij} - \sum_{j\in\Gamma^-(i)} y_{ji} = 0, \quad i \in V-\{0\} \tag{IV.15}$$

$$\sum_{j\in\Gamma^+(0)} y_{0j} \geq 1 \tag{IV.16}$$

$$0 \leq x_{ij}^k \leq Q^k y_{ij}, \quad \forall (i,j) \in A,\ i \in V-\{0\}, k \in K \tag{IV.17}$$

$$y_{ij} \in \{0,1\},\ x_{ij}^k \geq 0,\ u_j^k \geq 0,\ \forall\, i,j \in N,\ \forall\, k \in K \tag{IV.18}$$

As restrições IV.12-IV.14 ($x_{ij}^k \geq 0$) para cada $k \in K$, estão associadas a um problema de circulação de fluxo em G, o fluxo é oriundo de um sub-conjunto de nós de $V-\{0\}$ e a chegada em 0 com o valor igual a Q^k. Supõe-se que o problema tenha solução, ou seja: $Q^k \leq \sum_{i\in V-\{0\}} q_i^k$.

As restrições de IV.15-IV.18 formarão percursos viáveis para o veículo. Sabendo-se que $c_{ij} > 0$ então não haverá percursos na solução que não incluam o nó (0) de partida.

As restrições IV.13 podem também ser consideradas em sua forma agregada:

$$x_{ij}^k, \geq 0, \ (i,j) \in A,\ i \neq 0,\ k \in K$$

$$\sum_{k\in K} x_{ij}^k \leq (\sum_{k\in K} Q^k) y_{ij}, \quad (i,j) \in A,\ i \neq 0 \tag{IV.19}$$

Seja (P_1) o problema (P) ao ser substituida IV.17 por IV.18, o valor de $z(P)$[1] $= z(P_1) \geq z(\overline{P}) \geq z(\overline{P_1})$, onde $(\overline{P})$ e $(\overline{P_1})$ são as relaxações lineares de (P) e (P_1), e valor (.) é o valor ótimo da função objetivo de (.).

O primeiro modelo matemático (PC-1) acima proposto sugere que o PC é um problema do tipo NP-Árduo, uma vez que pode ser transformado ao problema do Caixeiro Viajante haja vista o percurso a ser realizado entre os centros de consumo, [Lenstra & Rinooy Kan 81]. O segundo modelo (PC-2), apesar das relaxações lineares, ainda sugere uma dificuldade quanto ao número de variáveis binárias existentes, e o número de restrições. De qualquer modo este modelo pode ser usado para instâncias razoáveis do PC (ou mesmo do PCV), desde que a matriz

[1] Valor da solução do problema P.

de custos não possua os ($c_{ij} > 0$, $\forall i \neq j$). No modelo PC-2 algum trabalho adicional deve ser realizado com o objetivo de tratar a relaxação do problema.

IV.1.2 Um Algoritmo Dinâmico de Compras

Em [Maculan et al 92] foi discutido um algoritmo dinâmico para o Problema do Viajante de Compras (**PVC**). Com o objetivo de tratar o PVC, os autores realizaram uma formulação de Programação Dinâmica, onde os estados são os centros de consumo, e os estágios os produtos a serem consumidos. Pertencem a cada estágio, apenas os estados que estão disponíveis para cada estágio. Nos estágios inicial e final, existem apenas o centro de consumo cc_0, figura IV.1. Assim, dada uma sequência de compra de produtos, a rota de compra entre os estágios inicial e final definirá a rota ótima de compra da sequência.

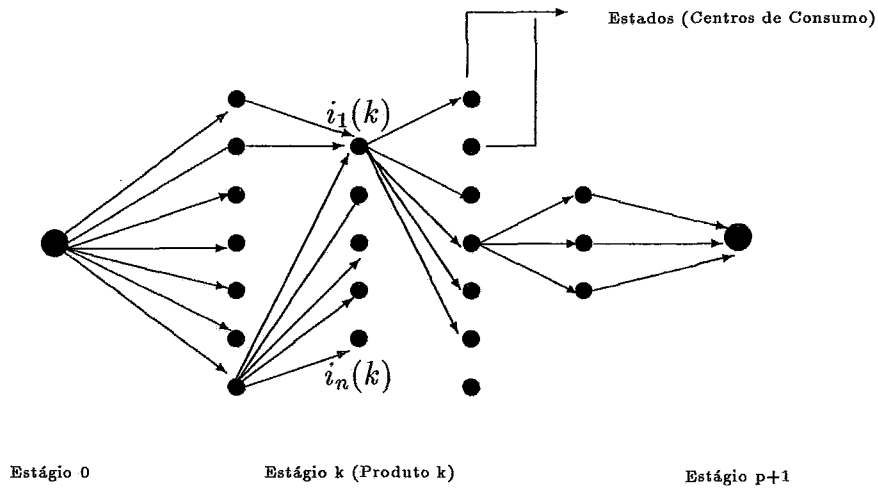


Figura IV.1: Formulação Dinâmica do Problema do Viajante de Compras.

Apesar de ser ótima a rota de compra para uma dada sequência, mostrou-se também que haviam $p!$ sequências admissíveis, uma das quais garantiria a rota ótima do **PVC**. No entanto, havia a possibilidade de se obter uma solução muito boa, ótima para uma sequência, em tempo $O(p\ n^2)$. Este resultado expressivo, torna tratável o **PVC**, e nos permite obter uma rotina para encontrar soluções viáveis para o Problema de Compras, como segue.

Algoritmo Dinâmico para o Problema do Viajante de Compras

Passo 0: A partir dos produtos disponíveis em cada centro, definir uma sequência de compra – Seq[], por produto, com os m produtos desejados;

Passo 1: Definir como estágio *zero* o centro origem, e como estágio $p+1$ o centro destino.

Construir um grafo direcionado e acíclico, onde os vértices dos estágios intermediários serão os centros de consumo que possuem o produto $j \in Seq[]$.

No grafo somente há ligação entre os vértices do estágio anterior e posterior, cujo custo de cada um será a distância entre os centros de compra considerados. Criar rótulos $Ct[i(j)]$ – Custo total dos produtos já adquiridos mais a distância total percorrida de s a $i(j)$ e o custo de compra do produto em $i(j)$.

Inicializar os rótulos $Ct[i(j)]$ e $Pred[i(j)]$ colocando os valores ∞ e -1, respectivamente. Já para o vértice do estágio inicial, fazer $Ct[s] = 0$.

Passo 2: Cálculo da Rota de Aquisição:

1. No estágio 1 fazer:

$$Ct[i(1)] \leftarrow c_{s,i(1)} + d_{i(1)};$$

$$Pred[i(1)] \leftarrow s$$

, onde s é o vértice origem.

2. Do estágio $i := 2$ até p, fazer:
Fazer para todo $j(i) \in Seq[]$

$$Ct[j(i)] \leftarrow \text{Min}\{c_{k(i-1),j(i)} + d_{j(i)} + Ct[k(i-1)]\}, \ \forall\, k \in Seq[i-1];$$

$$Pred[j(i)] \leftarrow k,$$

do valor mínimo encontrado.

3. No estágio $i := p+1$ fazer:

$$Ct[t] \leftarrow \text{Min}\{d_{k(i),t} + Ct[k(i-1)]\}, \ \forall\, k \in Seq[i-1];$$

$$Pred[t(i)] \leftarrow k(i-1),$$

é o predecessor do valor mínimo encontrado, e t é o vértice destino.

Passo 3: Construir a rota de compra, utilizando a lista de predecessores de $p+1$ a 1. Identificar os centros que foram utilizados na compra e os separar em um único vetor de rotas com elementos distintos. Armazenar o resultado dos locais e produtos comprados, e refinar a rota de compra utilizando algum algoritmo do Caixeiro Viajante ou do Caminho Hamiltoniano. Atualizar os valores totais da aquisição. Terminar.

No algoritmo dinâmico para o Problema do Viajante de Compras (PVC), observa-se que o custo total de compra está baseado numa compra equilibrada de apenas uma unidade de um dado produto de uma sequência. A princípio, pode-se aproveitar esta idéia e transformá-la em uma rotina para o Problema de Compras, procedendo com a qualificação dos produtos em cada centro de compra, baseado na mesma referência de disponibilidade de um dado produto em todos os centros que o dispõe. Deste modo não se privilegia nenhum centro de compra para quaisquer dos produtos. Já para a rota, mantém-se também as mesmas referências de início e fim da tarefa (Vértices origem e destino).

Supondo-se, então, que colocando o PC, onde agora existe um vetor de quantidades por produto nos centros, e um outro com a quantidade desejada por produto. Caso se deseje conhecer qual o melhor modo de comprar, deve-se permitir que todo produto seja avaliado pelo mínimo entre as quantidades mínimas disponíveis em todas as lojas, e a quantidade que ainda resta a ser adquirida. Procedendo de forma evolutiva, continuar assim até que não haja sobras em nenhuma das sequências, permitindo-se enfim saber como foi feita a compra, onde e quanto custou.

Em resumo, o algoritmo para o Problema de Compras (PC) pode ser assim descrito:

Passo 0: A partir dos produtos desejados, verificar se a soma dos produtos disponíveis em cada centro satisfaz a demanda das quantidades solicitadas, definir um vetor de sequência de compra – Seq[], que identifica os p produtos desejados;

Passo 1: Se as lojas não possuem no total as quantidades desejadas por produto, terminar. Senão, fazer até que a soma das sobras seja nula:

- Colocar em um vetor de resto a diferença da sobra por produto relativo ao adquirido. Atualizar os custos relativos por produto, baseado nas quantidades a serem adquiridas. Atualizar os custos relativos por produto, com base nas quantidades a serem adquiridas.
- Proceder os passos 1 e 2 do algoritmo dinâmico do problema do viajante de compras.

Passo 2: Construir a rota de compra, utilizando a lista de predecessores de $p + 1$ a 1. Identificar os centros que foram utilizados na compra e os separar em um único vetor de rotas com elementos distintos. Armazenar o resultado dos locais e produtos comprados, e refinar a rota de compra utilizando algum algoritmo do Caixeiro Viajante ou do Caminho Hamiltoniano. Atualizar os valores totais da aquisição. Permutar $Seq[]$ ou Terminar.

Como as quantidades e referências analizadas para cada sequência do PVC que é chamado no algoritmo do Problema de Compras, são as mesmas para

todo centro de compra, então para essa sequência, toda rota de compra gerada é a rota ótima da sequência para o Problema de Compras. Caso se deseje encontrar a solução ótima do problema global, basta repetir as sequências de compra possíveis até obter a menor solução, ou seja, após $p!$ iterações.

Com relação à complexidade computacional do algoritmo para o Problema de Compras, pode-se inicialmente admitir que todos os produtos têm custos e quantidades iguais nos n centros de consumo. Seja q a quantidade solicitada para cada centro de consumo e p o número de produtos a adquirir, então o algoritmo para o **PC** deverá perfazer

$$\sum_{k=1}^{n} p\, k^2 \ ,$$

ou seja, $O(p\, n^3)$ operações.

Como foi colocado, a obtenção de uma solução ótima para uma dada sequência de compra do **PC** é polinomial. Este resultado dá um certo ânimo ao fazer conjecturas sobre como obter a solução ótima do problema, utilizando poucas iterações, ao invés de se tentar calcular todas as $p!$ aqui descritas. Uma estratégia poderia ser o conhecimento ponderado da contribuição de cada item na formação da rota de compra. Outra estratégia poderia surgir na seleção da sequência de compra mais expressiva, uma última, e mais interessante seria de verificar na rota do caixeiro se há produtos com preços mais acessíveis do que aqueles escolhidos na sequência. Esta última conjectura foi utilizada, e em todas as avaliações que foram feitas, em poucas iterações se atinge resultados expressivos.

IV.1.3 Exemplo do Algoritmo Dinâmico

Considerando que a tabela IV.1 nos dá os custos de deslocamento entre centros e entre os centros e a origem, a tabela IV.2 informa os preços e disponibilidades de produtos por centro, e as quantidades a adquirir. Deseja-se saber como se deve fazer a compra dos produtos desejados na sequência (A, B, C), com custo total de aquisição mínimo.

Tomando o vértice 5 como o vértice origem.

Seja,

$$M\{x\} = \ Min\{x_i, i = 1, \ldots, n \mid x_i \neq 0\}$$

definido como o menor valor x, diferente de zero, de um dado produto cujas quantidades estão disponíveis nos centros de consumo ou são desejadas.

Tabela IV.1: Custos ($) das distâncias entre os centros de consumo.

	-	80	43	108	48	1
		-	59	66	37	2
D =			-	96	68	3
				-	38	4
					-	5
	1	2	3	4	5	

Tabela IV.2: Preços e estoques nos centros de compras

Produtos	A		B		C	
Centros	$	Qte	$	Qte	$	Qte
1	120	13	145	15	12	3
2	100	10	150	6	18	8
3	110	18	135	12	20	36
4	112	23	138	10	12	4
Desejado		16		8		6

Temos:

1a. Iteração: Sequência de Compra: Seq[]=(A, B, C).

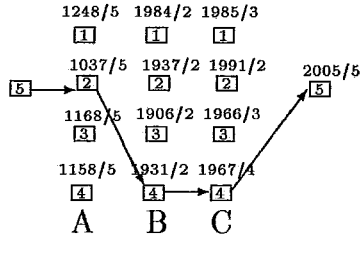
$$Qad[A] = M\{16, 13, 10, 18, 23\} = 10, \quad Resta[A] = 6$$

$$Qad[B] = M\{8, 15, 6, 12, 10\} = 6, \quad Resta[B] = 2$$

$$Qad[C] = M\{6, 3, 8, 36, 4\} = 3, \quad Resta[C] = 3$$

Prepare o Algoritmo Dinâmico do Problema do Viajante de Compras.

Produto Loja	A $	B $	C $
1	1200	1370	36
2	1000	900	54
3	1100	810	60
4	1120	820	36



Loja[2].Adquire[A].Qte = 10
Loja[4].Adquire[B].Qte = 6
Loja[4].Adquire[C].Qte = 3

$$\sum Resta[] \ > \ 0$$

, Então Retorne ao Passo 1;

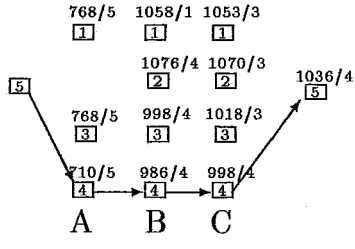
$$Qad[A] \ = \ M\{6,13,0,18,23\} \ = \ 6, \quad Resta[A] = 0$$

$$Qad[B] \ = \ M\{2,15,6,12,4\} \ = \ 2, \quad Resta[B] = 0$$

$$Qad[C] \ = \ M\{6,3,8,36,4\} \ = \ 3, \quad Resta[C] = 2$$

Prepare o Algoritmo Dinâmico do Problema do Viajante de Compras.

Produto Loja	A $	B $	C $
1	720	290	12
2	-	300	18
3	660	270	20
4	672	276	12



Loja[4].Adquire[A].Qte = 6
Loja[4].Adquire[B].Qte = 8
Loja[4].Adquire[C].Qte = 4

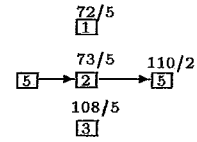
$$\sum Resta[] \ > \ 0$$

, Então Retorne ao Passo 1;

$$Qad[C] \ = \ M\{2,3,8,36,0\} \ = \ 2, \quad Resta[C] = 0$$

Prepare o Algoritmo Dinâmico do Problema do Viajante de Compras.

Produto Loja	C $
1	24
2	36
3	40
4	-



C

Loja[2].Adquire[C].Qte = 2

$$\sum Resta[] = 0$$

, Então Termine e Gere uma Rota refinada com os locais de compra.

<u>Resultado:</u>

- Rota de Aquisição: 5 - 2 - 4 - 5
- Custo da Rota: 141
- Custo de Compra: Loja 2: 1036 (10-A,2-C);
  Loja 4: 1824 (6-A,8-B,4-C) → Total: 2860
- Custo Total de Aquisição e Compra: 3001.

IV.1.4 Avaliação Computacional

O algoritmo Dinâmico do **PC**, foi implementado no compilador Turbo Pascal V6.0 –, Borland International, em ambiente MS-DOS num microcumputador PC-AT/386 SX 16MHz, utilizando os recursos gráficos do sistema **SisGRAFO**.

Na tabela IV.3 são discriminados os problemas e os tipos de grafos (G) utilizados. Considerou-se como tipo 0, um grafo que utiliza coordenadas Euclidianas, [Campello & Maculan 94]; do tipo 1 um grafo com matriz de distâncias simétrica e do tipo 2 um grafo com matriz de distâncias assimétrica.

Considerar :

- (Mod) é o modelo básico do grafo que suporta o problema proposto;
- (Og) e (Dt) como os vértices inicial e final da rota, respectivamente;
- n o número de centros de compra e p o número de produtos a comprar;
- (NS) o número de centros de compra na rota;
- (α) um multiplicador de custo por unidade de distância;
- ($ Compra) o custo total de compra e ($ Total) é a soma do custo de compra com o custo de percurso e
- Tempo - Tempo de processamento, (hora : minuto : segundo . décimo de segundo).

Foram utilizados os algoritmos de [Dijkstra 59] do caminho mínimo e B&B [Syslo et al 83], para o Problema do Caixeiro Viajante durante as fases inicial e final do algoritmo, respectivamente.

Dentro das intâncias analizadas na tabela IV.3, os modelos de 1 a 7 são grafos padrões que mantêm sua mesma estrutura básica (posição de centros de consumo, custos e disponibilidades por centro). Os problemas, são instâncias do modelo. Veja-se por exemplo o modelo 1, este tem os problemas 1, 2 e 3, cujas instâncias são respectivamente iguais com relação ao modelo, porém o que muda de um em relação aos outros é o peso dado ao custo de distância, associado diretamente à função objetivo. As instâncias 1.1, 2.1 e 3.1, e as demais também são iguais seguindo o mesmo raciocínio anteriormente exposto. Nos modelos de 2 a 7 o mesmo acontece às instâncias relativas aos problemas particulares.

O modelo 1, tabela IV.3, representa uma situação real (custos de compra) de parte de uma cidade. Este modelo foi propositadamente colocado haja vista uma observação mais criteriosa do problema básico de formação da rota de compra a partir de uma única lista de aquisição.

As variações dos custos de aquisição e totais foram consideráveis no modelo 2. Nota-se na tabela IV.3 a significativa redução ocorrida no custo de compra nos problemas 4, 5, 6 e 7, e a diminuição no número de centros de compra visitados entre as soluções que dão mais valor ao custo de distância, problemas 4 e 5 e as que dão menos valor, problemas 6 e 7. Este fato permitiu com que se visualizasse a presença de um ponto de partida ou local ótimo de partida ligado ao PC, gerando uma extensão interessante a ser pesquisada.

Na tabela IV.3, nota-se enfim que para um mesmo problema houve a influência decisiva do custo de distância na formação da rota dos centros de compra a serem escolhidos. Esta influência depende exclusivamente da formação dos preços

Tabela IV.3: Análise do comportamento do Algoritmo Dinâmico para o PC.

Mod	Prob	n	p	G	Og	Dt	NS	α	$ Compra	$ Total	Tempo
	1	7	24	0	1	1	3	1,00	111.814,00	112.321,00	25.87
	1.1	7	24	0	1	1	3	1,00	111.143,00	112.029,50	27.02
	1.2	7	24	0	1	1	3	1,00	111.814,00	112.787,00	25.59
	1.3	7	24	0	1	1	3	1,00	111.811,00	113.130,00	26.14
	1.4	7	24	0	1	1	3	1,00	111.814,00	112.983,00	25.70
1	2	7	24	0	1	1	4	0,50	111.143,00	111.915,50	25.16
	2.1	7	24	0	1	1	3	0,50	111.814,00	112.902,00	26.31
	2.2	7	24	0	1	1	4	0,50	111.143,00	111.912,50	26.09
	2.3	7	24	0	1	1	5	0,50	111.140,00	112.019,00	26.48
	2.4	7	24	0	1	1	4	0,50	111.143,00	112.244,00	25.98
	3	7	24	0	1	1	5	0,10	110.404,00	110.647,60	31.36
	3.1	7	24	0	1	1	5	0,10	110.404,00	110.650,30	26.63
	3.2	7	24	0	1	1	5	0,10	110.404,00	110.646,80	26.58
	3.3	7	24	0	1	1	6	0,10	110.401,00	110.669,70	27.18
	3.4	7	24	0	1	1	5	0,10	110.404,00	110.682,40	27.69
	4	11	6	0	10	11	7	3.00	1.226,00	4.901,00	51.39
	5	11	6	0	10	11	7	1,00	1.220,00	2.340,00	51.69
	5.1	11	6	0	10	11	8	1,00	1.206,00	2.168,00	53.66
	5.2	11	6	0	10	11	4	1,00	1.227,00	2.308,00	51.46
	5.3	11	6	0	10	11	6	1,00	1.067,00	2.046,00	1:05.20
	5.4	11	6	0	10	11	7	1,00	1.034,00	2.097,00	1:04.10
2	6	11	6	0	10	11	6	0.50	698,00	1.277,00	38.50
	7	11	6	0	10	11	6	0,10	488,00	612,10	37.68
	7.1	11	6	0	10	11	6	0,10	1.027,00	1.122,70	51.13
	7.2	11	6	0	10	11	7	0,10	1 108,00	1.216,10	49.59
	7.3	11	6	0	10	11	9	0,10	922,00	1.038,80	51.52
	7.4	11	6	0	10	11	7	0,10	923,00	1.029,30	49.32
	8	21	10	0	13	13	15	1,00	2.281.921,00	2.283.294,00	25:02.44
3	9	21	10	0	13	13	15	0,50	2.278.973,00	2.279.659,50	25:00.29
	10	21	10	0	13	13	14	0,10	2.262.197,00	2.262.212,75	25:02.54
4	11	5	3	1	5	5	2	1,00	2.860,00	3.001,00	4.01
	12	15	4	1	1	1	8	3,00	15.289,00	15.514,00	55.58
	13	15	4	1	1	1	9	1,00	15.098,62	15.184,62	1:01.79
5	14	15	4	1	1	1	9	1,00	15.090,70	15.176,70	(*) 3:39.75
	15	15	4	1	1	1	7	0,50	14.966,18	15.003,68	1:05.36
	16	15	4	1	1	1	8	0,10	14.960,20	14.970,10	1:05.13
6	17	54	8	2	62	31	10	1,00	267,24	394,24	(*) 1:29:41.27
7	18	35	5	2	19	55	9	1,00	529,70	634,70	25:40.50

(*) Problema executado em mais de uma iteração.

dos produtos e de suas quantidades em cada loja. A maior dispersão dos valores pode caracterizar soluções bastante incomuns como se pode notar.

O modelo 3 já mostra como um caso patológico na formação de rotas pode ocorrer quando muitas composições de compra precisam ser feitas em vários centros. Este exemplo foi elaborado como uma maneira de mostrar a complexidade do PC em relação à dispersões de quantidades de produto, como foi descrito anteriormente.

Nos modelos de 4 a 7, verifica-se situações em grafos simétricos e mistos, com pouca densidade de arcos/elos por vértice, porém com vértices espalhados pela rede em pontos específicos de um grafo. Situações semelhantes são encontradas no mundo real, quando se tem uma opção de rede de distribuição utilizando ruas e avenidas de uma cidade.

As figuras IV.2, IV.3 e IV.4 mostram algumas das instâncias avaliadas a partir das soluções geradas com o sistema **SisGRAFO**.

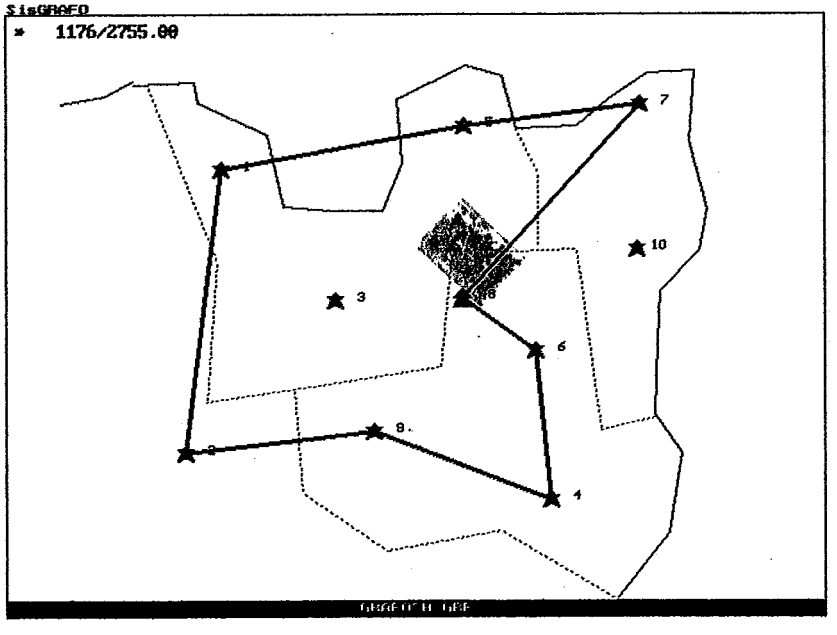


Figura IV.2: Uma solução do PC em um grafo Euclidiano.

Extensões do PC podem advir como consequência. Pode-se tê-lo com restrições no custo total de aquisição, limitações na distância total a ser percorrida, com janelas de tempo nos centros de compra –, [Ochi et al 92], localização da origem, e uma extensão maior como sendo os *m-Compradores* o qual considera que

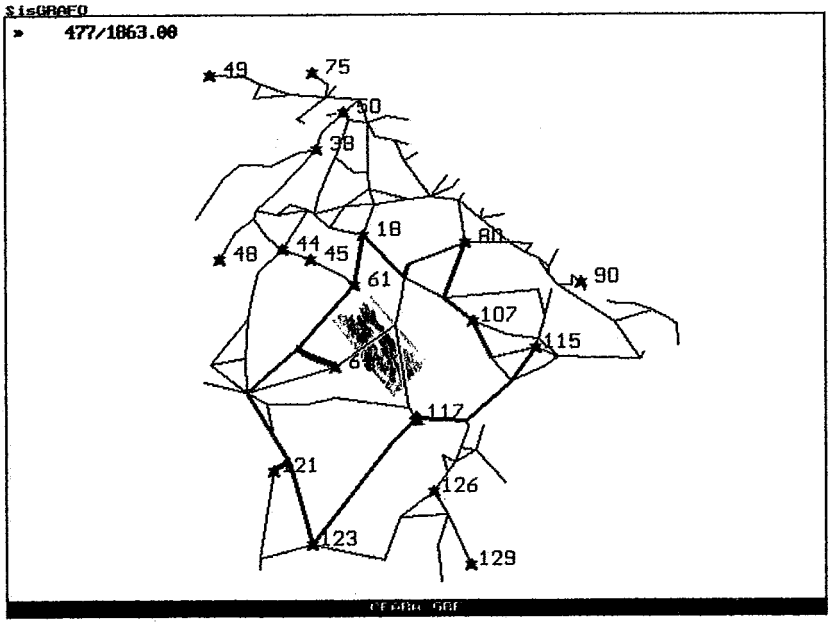


Figura IV.3: Uma solução do PC em grafo Simétrico.

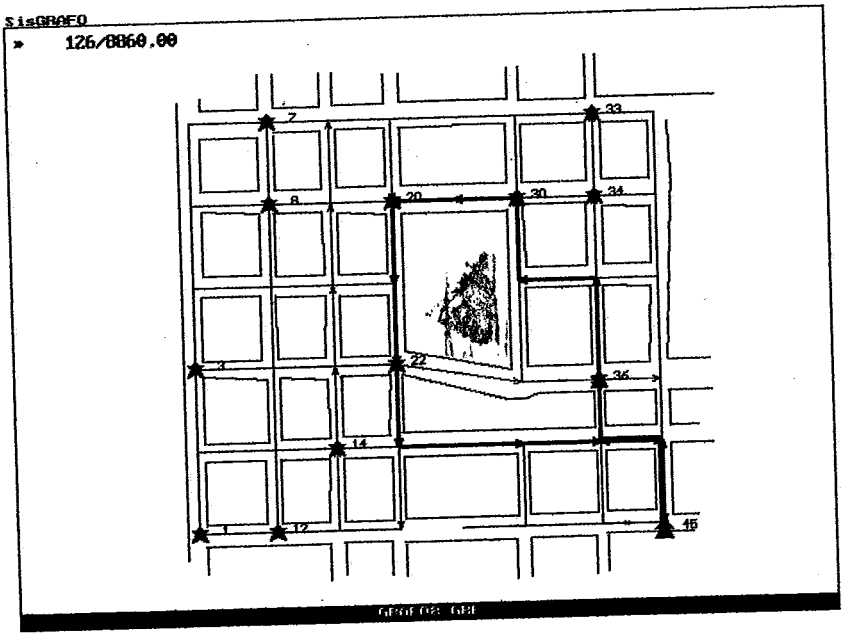


Figura IV.4: Uma solução do PC em grafo Misto.

1-*Comprador* possui capacidade limitada para compor uma compra (Capacidade do Veículo), onde cada produto possui um peso por unidade –, um problema de compras com restrições do problema das múltiplas mochilas, [Martello & Toth 90]. Outras extensões também podem surgir, as quais estariam mais próximas da realidade do processo de compra para um contexto específico.

Uma observação importante sobre o PC é a possibilidade de colocá-lo como um problema de vendas, ao invés de compras. Neste caso, a venda constitui-se no faturamento. Para esta extensão, temos que o problema será de maximizar o lucro na venda dos produtos. A grande vantagem desta abordagem está com relação à proximidade com a realidade, principalmente se considerarmos uma ocorrência estocástica no processo de venda ou mesmo determinística, caso haja um esquema de pré-venda já amplamente estabelecido na aplicação real, como por exemplo no roteamento de veículos vendedores de massas e biscoitos.

Capítulo V

O Problema de Percurso de Veículos em Vértices

O Problema de Percurso de Veículos em Vértices (PPV), pode ser descrito como o problema de determinar um conjunto ótimo de rotas de entrega/coleta a partir de um ou múltiplos depósitos, sobre um número de clientes dispersamente distribuidos em uma região geográfica, sujeito a um conjunto de restrições laterais (demanda máxima conduzida nos veículos, tempo de serviço, distância máxima, horários de atendimento, etc.), figura V.1 e V.2.

Este problema foi inicialmente formulado por [Dantzig & Ramser 59], que o chamaram de Problema de Despacho de Caminhões. O objetivo era resolver um problema de distribuição de derivados de petróleo em postos. No trabalho que desenvolveram, eles apresentaram um método heurístico que basicamente estava mais preocupado com o preenchimento dos veículos de igual capacidade (frota homogênea) do que com a distância percorrida.

[Clarke & Write 64] apresentaram um método bem mais eficiente, o qual é hoje conhecido como *Método de Economias.* Os autores exploraram a técnica aplicando-a a uma extensão do problema onde os veículos poderiam ter diferentes capacidades (Frota Heterogênea). A simplicidade do método e a facilidade de implementá-lo, tornou-o a base de grande parte dos softwares relacionados a esta aplicação, [Golden et al 86], [Belardo et al 82], [Golden & Wasil 87]. Apesar de na grande maioria das instâncias o método manter-se longe da solução ótima, a facilidade de inserção do tratamento de restrições adicionais dão a ele uma grande versatilidade no encaminhamento da solução de problemas especiais ligados ao PPV. Muito esforço, porém, vem sendo empreendido na melhoria do método das economias, alguns já têm logrado uma efetiva competitividade com os melhores métodos já elaborados, [Paessens 88], [Negreiros 93].

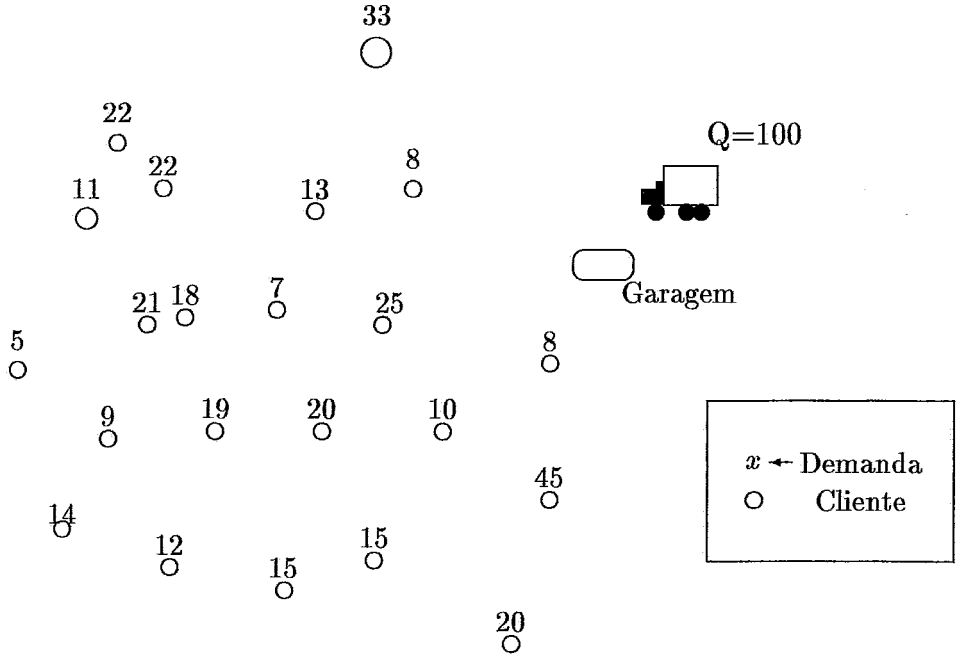


Figura V.1: Situação geográfica de distribuição para Clientes e suas Demandas.

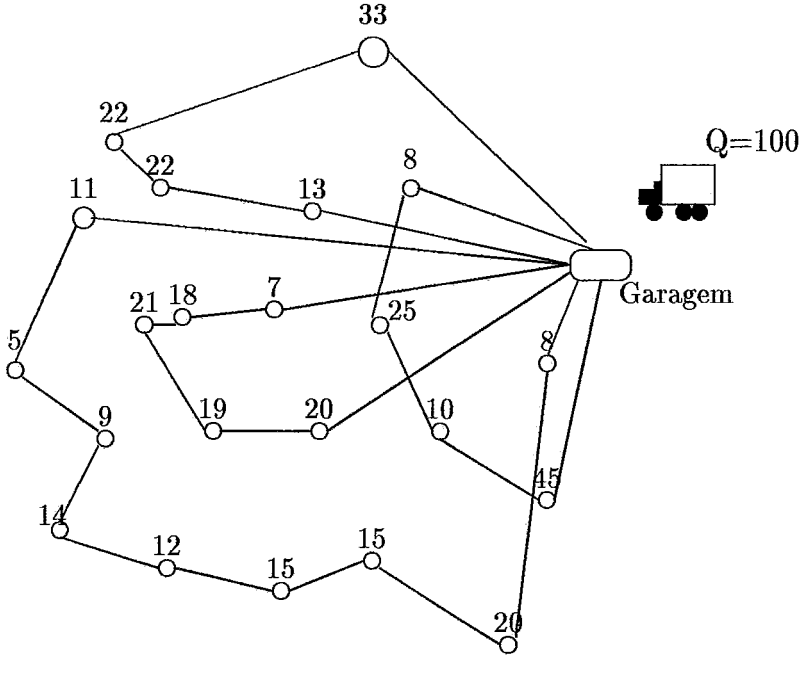


Figura V.2: Uma solução do PPV aplicado à figura anterior.

O PPV possui um conjunto muito grande de abordagens, e hoje tem-se mostrado como uma aplicação de sucesso em Pesquisa Operacional, justificando a grande soma de publicações existentes, [Bodin et al 83], [Golden & Assad 86], [Golden & Assad 88], [Golden & Wong 92].

O PPV nasce basicamente de uma particularização do problema do m-Caixeiro Viajante (m-PCV), onde se deseja encontrar o percurso mínimo total para m caixeiros. O m-PCV pode ser formulado como segue.

Sejam dados:

c_{ij}, o custo de atravessar uma ligação de i a j;

M, o número de caixeiros;

V, o conjunto formado pelos vértices que serão visitados;

$|V| = n$, o número de vértices

$$x_{ij} = \begin{cases} 1, & \text{Se a ligação de } i \text{ a } j \text{ é usada;} \\ 0, & \text{Caso contrário.} \end{cases}$$

S, o conjunto formado pelas restrições de quebras de subrotas ilegais que não incluem o vértice origem.

$$(m\text{-PCV}) \text{ Minimizar} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{V.1}$$

Sujeito a:

$$\sum_i x_{ij} = b_j = \begin{cases} M, & \text{Se } j = 1; \\ 1, & \text{Caso contrário} \end{cases} \tag{V.2}$$

$$\sum_j x_{ij} = a_i = \begin{cases} M, & \text{Se } i = 1; \\ 1, & \text{Caso contrário} \end{cases} \tag{V.3}$$

$$X = (x_{ij}) \in S \tag{V.4}$$

$$x_{ij} \in \{0,1\} \quad (i,j = 1,\ldots,n) \tag{V.5}$$

As restrições V.2-V.3 requerem que todos os M caixeiros sejam usados, saindo e chegando no vértice inicial 1. As restrições V.3 garantem a não formação de subrotas ilegais que não incluam o vértice origem 1. As restrições V.5 definem as variáveis de decisão do modelo.

A generalização aparente que o m-PCV apresenta é ilusória, uma vez que todos os problemas deste tipo podem ser convertidos em um PCV equivalente. Estes recursos estão descritos mais detalhadamente na seção V.1 a seguir.

Nas seções a seguir discute-se as mais importantes aplicações sobre o PPV e suas variações. O problema genérico, capacitado e/ou restrito em tempo/distância é inicialmente discutido na seção V.1 considerando alguns dos métodos exatos e heurísticos propostos pela literatura. Na seção V.2 tem-se uma discussão sobre o PPV com múltiplos depósitos centrais, problema que vem tendo destaque através de métodos exatos de geração de colunas. Na seção V.3, tem-se uma discussão sobre o PPV com Janelas de Tempo. Na seção V.4 tem-se o PPV com *Repuxo de Carga*-(*Backhaul*), uma aplicação advinda do transporte de carga em longas distâncias. Na seção V.5 tem-se sobre o PPV com Estoque ou com Atendimento Periódico, a partir de demandas não determinísticas. Na seção V.6 tem-se o Problema de Percurso com composição de frota, onde neste caso considera-se o problema de formação de uma frota considerando a importância do custo fixo na utilização de veículos distintos, próprios ou afretados. A seção V.7 considera o PPV com a possibilidade de divisão de demanda, ou seja o serviço a um cliente pode ser realizado por mais de um veículo. Por fim, na seção V.8, tem-se o PPV aplicado a situações onde os veículos são equipados com múltiplos compartimentos de carga, onde se discute um método de particionamento elástico e uma aplicação com atribuição múltipla de clientes a um compartimento.

V.1 O PPV Genérico

Como foi dito, o PPV requer um conjunto de rotas partindo de um depósito central a vários pontos de demanda, cada um deles tendo necessidades de serviço que deverão ser integralmente atendidas pela mínima distância total coberta por uma frota. Os veículos são restritos à sua capacidade (volume ou peso de carga), e em alguns casos a tempos ou distâncias máximas de percursos com partida de um único depósito central. A este problema denomina-se PPV Genérico (PPV-G), [Golden et al 84a].

No caso em que as restrições de tempo por percurso ou de capacidade são omitidas, tem-se o problema padrão de percurso de veículos em nós. Como está proposto, o problema trata de uma atividade de apenas entrega e/ou coleta de um ou vários produtos onde a única preocupação é o espaço que ocupam ou o seu peso.

Uma formulação para PPV-G, pode ser colocada como segue, conforme [Golden et al 84a].

Sejam dados,

n, o número de vértices com demanda incluindo o depósito (1);

nv, o número de veículos;

Q^v, a capacidade do veículo v;

q_i, a demanda no vértice i, onde: $q_1 = 0$ e $q_i \leq Q^v$, $\forall\, i, v$;

T^v, o tempo máximo permitido para o veículo v;

t_i^v, o tempo necessário para o veículo v entregar ou receber no vértice i (tempo de serviço em i por v);

t_{ij}^v, o tempo de percurso do veículo v entre os vértices i e j, $t_{ii}^v = \infty$

$$x_{ij}^v = \begin{cases} 1, & \text{Se a ligação } ij \text{ é usada pelo veículo } v \\ 0, & \text{Caso contrário.} \end{cases}$$

S, o conjunto formado pelas restrições de quebras de subrotas ilegais que não incluem o vértice origem.

$$\text{(PPV-G) Minimizar} \sum_{v=1}^{nv} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}^v \tag{V.6}$$

Sujeito a:

$$\sum_v \sum_i x_{ij}^v = 1 \quad (j = 2, \ldots, n) \tag{V.7}$$

$$\sum_v \sum_j x_{ij}^v = 1 \quad (i = 2, \ldots, n) \tag{V.8}$$

$$\sum_i x_{ip}^v - \sum_j x_{pj}^v = 0 \; (p = 1, \ldots, n) \; (v = 1, \ldots, nv) \tag{V.9}$$

$$\sum_i q_i (\sum_j x_{ij}^v) \leq Q^v \quad (v = 1, \ldots, nv) \tag{V.10}$$

$$\sum_i t_i^v \sum_j x_{ij}^v + \sum_i \sum_j t_{ij}^v x_{ij}^v \leq T^v \quad (v = 1, \ldots, nv) \tag{V.11}$$

$$\sum_i x_{i1}^v \leq 1 \; (v = 1, \ldots, nv) \tag{V.12}$$

$$\sum_j x_{1j}^v \leq 1 \; (v = 1, \ldots, nv) \tag{V.13}$$

$$X = (x_{ij}^v) \in S \tag{V.14}$$

$$x_{ij}^v \in \{0, 1\} \quad (i, j = 1, \ldots, n) \; (v = 1, \ldots, nv) \tag{V.15}$$

A função objetivo deseja minimizar o custo total do percurso de um conjunto de veículos iguais ou distintos. As restrições V.7-V.8 definem que cada vértice demanda é servido por exatamente um veículo. As equações V.9 consideram a continuidade do percurso (rota), ou seja, se um veículo entra em um vértice com demanda ele deve sair. As equações V.10 são as restrições de capacidade dos veículos, de modo similar as equações V.11 são relativas ao tempo total da rota. As equações de V.12-V.13 garantem que a disponibilidade de veículos não será excedida. Por fim, as restrições V.14 representam a proibição de subrotas ilegais. No PPV-G, uma vez que um cliente solicita serviço, sua demanda é plenamente atendida, ou melhor, não se permite o atendimento parcial de qualquer demanda.

Nota-se do modelo que quando as restrições de capacidade V.10 e de tempo V.11 não têm importância, tem-se uma redução ao m-PCV, como pode ser visto pela eliminação das restrições V.8 e substituindo-as por $x_{ij} = \sum_v x_{ij}^v$ na função objetivo e restrições reminiscentes.

Pode-se, sem dificuldade, generalizar mais ainda o modelo genérico para o caso de múltiplos produtos, onde muitos tipos de produtos podem ser enviados simultaneamente sobre uma rede, de modo a satisfazer as demandas individuais de vários clientes. Outras restrições também podem ser adicionadas ao modelo, como as restrições de múltiplos depósitos, janelas de tempo, prioridades no atendimento (precedência), dentre outras, algumas das quais serão discutidas nas seções seguintes.

No tratamento do PPV Genérico, uma gama de métodos exatos e heurísticos vêm sendo discutidos pela literatura, sendo que um maior destaque está sendo dado nos últimos anos aos métodos heurísticos baseados em busca locais meta-heurísticas, [Hjorring 95]. Um grande esforço, porém vem sendo feito na melhoria de desempenho dos métodos exatos, destacando-se principalmente o trabalho de [Fisher 94].

Nas subseções a seguir, discute-se alguns dos métodos exatos e heurísticos usados no tratamento do PPV-G conforme classificação proposta por [Laporte & Nobert 87], e revisada em [Laporte 92].

V.1.1 Algoritmos Exatos do PPV Genérico

Os algoritmos exatos do PPV são classificados como, [Laporte & Nobert 87]:

1. Métodos de Busca Direta em Árvore;
2. Métodos de Programação Dinâmica e
3. Métodos de Programação Linear.

Uma vez que são muitos os enfoques sobre os métodos, considera-se a seguir alguns daqueles que se mostraram mais eficientes, considerando o ponto de vista da qualidade das soluções (distância do ótimo), tempo de processamento e tamanho das instâncias analisadas.

Métodos de Busca Direta em Árvore

Dois algoritmos serão considerados, o primeiro será o algoritmo proposto para o caso de um PPV padrão considerando a atribuição de Limites Inferiores (LI) e *B&B*, o qual obteve excelente desempenho para instâncias assimétricas significativamente grandes, [Laporte & Nobert 87]. O segundo método, mais recente, k-Árvore mostrou-se forte no tratamento das soluções do PPV simétrico padrão, na maioria das instâncias propostas pela literatura, [Fisher 94].

Algoritmo de Atribuição LI e B&B

Este algoritmo explora a relação existente entre o PPV padrão (com restrição de capacidade ou de tempo) e uma de suas relaxações, o m-PCV. Onde dado um grafo $G = (V, A)$ com o vértice depósito (1) e m veículos nele baseados, o m-PCV, como foi visto, consiste na geração de m-rotas de custo mínimo iniciando e terminando no depósito, de modo que todo vértice remanescente é visitado exatamente uma vez. Pela transformação do m-PCV em 1-PCV, mostra-se que o problema m-PCV é do tipo NP-Árduo, [Lenstra & Rinooy Kan 81].

Dentre os passos necessários para se transformar o m-PCV em 1-PCV, como já foi colocado. Algumas regras, porém, são gerais para o tratamento do problema como um PPV via m-PCV, como seguem.

1. Aumentar o número de vértices pela introdução de $m-1$ depósitos. Seja $n' = n + m - 1$, $V' = \{1, \ldots, n'\}$ e $A' = A \cup \{< i, j > : \ i, j \in V', i \neq j, \ i \text{ ou } j \in V' \setminus V\}$.

2. Matriz de distâncias estendida associada a A':

$$c'_{ij} = \begin{cases} c_{ij} \ (i, j \in V) \\ c_{i1} \ (i \in V \setminus \{1\}, j \in V \setminus \{1\}) \\ c_{1j} \ (i \in V' \setminus V, j \in V \setminus \{1\}) \\ \alpha \ (i, j \in (V' \setminus V) \cup \{i\}) \end{cases}$$

onde α depende da variante do problema em consideração, ou seja:

$\alpha = \infty$, quando se considera o problema de exatos m-caixeiros;

$\alpha = 0$, quando se considera o problema para pelo menos m-caixeiros;

$\alpha = -\infty$, deseja-se conhecer através do problema o menor valor de m.

O PPV padrão pode então ser formulado como segue.

Sendo as variáveis:

$$x_{ij} = \begin{cases} 1, & \text{Se o arco } < i,j > \text{ é usado, para } i \neq j. \\ 0, & \text{Caso contrário.} \end{cases}$$

$q_i + q_j \geq Q$, e i e j não podem pertencer à mesma rota, ou seja, neste caso x_{ij} não é definido.

$$\text{(PCV-P1) Minimizar} \sum_{i=1}^{n}\sum_{j=1}^{n} c'_{ij}x_{ij} \tag{V.16}$$

Sujeito a:

$$\sum_{j} x_{ij} = 1 \quad (i = 1, \ldots, n') \tag{V.17}$$

$$\sum_{i} x_{ij} = 1 \quad (i = 1, \ldots, n') \tag{V.18}$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - v(S) \quad (S \subset V' \setminus \{1\} : |S| \geq 2) \tag{V.19}$$

$$x_{ij} \in \{0,1\} \quad (i,j = 1, \ldots, n') \; (i \neq j) \tag{V.20}$$

As restrições V.17-V.18 representam um problema de atribuição modificado (atribuição à diagonal principal é proibida). As restrições V.19 são aquelas referente à eliminação de subrotas ilegais, onde $v(S)$ é o limite inferior apropriado sobre o número de veículos necessários para visitar todos os vértices de S na solução ótima. Estas restrições são obtidas, observando-se que $\{S \subset V' \setminus \{1\}, |S| \geq 2, \overline{S} = V' \setminus V\}$, tem-se,

$$\sum_{i \in S}\sum_{j \in \overline{S}} x_{ij} \geq v(S)$$

e que se verifica a seguinte identidade,

$$|S| = \sum_{i,j \in S} x_{ij} + \sum_{i \in S}\sum_{j \in \overline{S}} x_{ij}.$$

O valor de $v(S)$ depende do tipo de PPV em consideração. No PPV padrão (capacitado), pode-se tomar um limite do problema de *binpacking*, ou

$$v(S) = \lceil \frac{\sum_{i \in S} q_i}{Q} \rceil$$

Em geral, não é fácil a determinação de um LI para o PPV *a priori*. Normalmente um LI é obtido na evolução de um algoritmo B&B. No caso onde não se consegue atingir uma solução, é válido assumir $v(S) = 1$. As restrições V.19 possuem pois uma função dual: Elas permitem que todas as rotas dos veículos satisfaçam às restrições de capacidade ou de tempo/distância máxima. Elas também garantem que a solução não contém subrotas desconexas do depósito, uma vez que todo subconjunto S de $V \setminus \{1\}$ está ligado a seu complemento.

Assim como o algoritmo baseado em atribuição para o PPV, o problema é resolvido através do procedimento B&B onde os sub-problemas gerados nas ramificações são Problemas de Atribuição. A única diferença é que surge na definição das subrotas ilegais os formatos de rotas permitidos pelo PPV padrão. São elas:

1. Subrotas sobre um conjunto S de vértices de $V \setminus \{1\}$;

2. Restrições sobre a violação de capacidade ou tempo/distância. Consistindo de caminhos de vértices $(i_1, i_2, \ldots, i_r)$ onde $i_1, i_r \in \{1, n+1, \ldots, n+m+1\}$, $i_2, \ldots, i_{r-1} \in V \setminus \{1\}$ e $\sum_{t=2}^{r-1} q_{it} > Q$ ou

$$\sum_{t=1}^{r-1} c_{i_t, i_{t+1}} > D.$$

   Sendo D a distância máxima permitida.

   Isto pode ser eliminado pelo particionamento do problema corrente inviável, tomando $v(S) = 1$. Outros esquemas de particionamento mais refinados podem ser considerados, [Laporte & Nobert 87].

Ao final do processo, será necessário interpretar a solução como um PPV. São então aplicadas as seguintes regras:

Considerar um arco $< i, j >$ pertencendo à solução.

1. Se $i \in V \setminus \{1\}$ e $j \in V' \setminus V$, traçar $< i, j >$ por $< i, 1 >$;

2. Se $i \in V' \setminus V$ e $j \in V' \setminus V$, traçar $< i, j >$ por $< 1, j >$;

3. Se $i, j \in V' \setminus V$, retirar $< i, j >$

[Laporte & Nobert 87] usaram a metodologia acima para resolver problemas assimétricos do PPV em instâncias aleatoriamente geradas com 100 a 260 vértices, e obtiveram a solução ótima em tempos que variaram entre 740 a 5119 segundos, em um computador VAX 11/780, usando um compilador FTN. Os autores também consideraram a execução do algoritmo sobre problemas simétricos,

mas como era de se esperar, os resultados mostraram-se pobres para instâncias com um número de vértices superiores a 30. Para instâncias de 10 a 30 vértices, foram atingidas soluções ótimas em tempos que variaram entre 1171,4 a 3223,7 segundos.

Algoritmo K-Árvores

O método proposto por [Fisher 94], é um avanço sobre tudo o que se tem feito até hoje em PPV padrões. O problema tratado considera o PPV Capacitado, porém não permitindo que um único cliente seja atendido por um único veículo.

Baseado nas idéias propostas por [Held & Karp 70] para o PCV, o método considera que sendo dado um grafo com n vértices, define-se um conjunto K-Árvore com grau $2K$ no depósito (1), juntamente a restrições de capacidade impostas pelo veículo, e a necessidade de que o grau de cada vértice (demanda) seja exatamente 2. As restrições laterais são dualizadas de modo a obter uma relaxação lagrangeana que é definida como o problema da Árvore de Grau Mínimo Restrita, ou da K-Árvore.

O modelo proposto por Fisher, pode ser descrito como segue.

Sejam dados,

$N_1 = V \setminus \{1\}$

$x_{ij} = \begin{cases} 1, & \text{Se o elo } (i,j) \text{ está na solução;} \\ 2, & \text{Caso contrário} \end{cases}$

Sendo G um grafo simétrico, então $x_{ij} = x_{ji}$. Toma-se $X = \{x\}$ a representação de uma K-Árvore com grau $2K$ no depósito. Ou seja,

$$x = \{x_{12}, x_{13}, \ldots, x_{1n}, x_{23}, \ldots, x_{n-1,n}\}$$

$$X = \{x | x \in \{0,1\}\}, \text{e definir uma } K\text{-Árvore satisfazendo a } \sum_{i=2}^{n} x_{1i} = 2K$$

Para $S \subseteq V$, seja $\overline{S} = V \setminus S$, $a(S) = \sum_{i \in S} a_i$ e $r(S) = \lceil \frac{a(S)}{b} \rceil$, onde $E(S)$ é definido como o conjunto de um grafo completo não direcionado sobre o conjunto de nós S, isto é, $E(S)$ é o conjunto de todos os pares ordenados ij, $i \in S, j \in S, i \neq j$.

$r(S)$, é o limite inferior sobre o número de veículos necessários ao serviço dos clientes no conjunto S.

$$\text{(PPV K-Árvore) Minimizar}_{x \in X} \sum_{j \in E(V)} c_{ij} x_{ij} \tag{V.21}$$

Sujeito a:

$$\sum_{j=1, j \neq i}^{n} x_{ij} = 2 \quad (i = 1, \ldots, n) \tag{V.22}$$

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} = 2r(S) \; \forall S \subseteq N_1, |S| \geq 2 \tag{V.23}$$

$$x_{ij} \in \{0, 1\} \quad (i, j = 1, \ldots, n) \; (i \neq j) \tag{V.24}$$

As restrições V.22-V.23 representam a continuidade da árvore e as restrições sobre os veículos, respectivamente. O autor incorporou estas restrições na função objetivo na forma de relaxação lagrangeana do PPV. Devido às $O(2^n)$ restrições no conjunto em V.23. Fisher resolve o problema pela geração dinâmica destas restrições à medida que são violadas (identificação de mudança de espaço no dual).

A relaxação lagrangeana é resolvida pelo autor usando otimização por subgradiente. Obtém-se com isto limites inferiores mais estreitos, bem melhores até do que os limites encontrados por [Christofides et al 81a]. Três heurísticas são usadas de modo a tomar a K-Árvore obtida de cada iteração do método de subgradiente, e transformá-la em uma solução viável do PPV. À medida que o LI da K-Árvore se aproxima de uma solução ótima, as heurísticas são capazes de encontrar LIs cada vez melhores.

Nos resultados apresentados sobre instâncias padrões da literatura e novas propostas pelo autor (entre 25 e 134 clientes), todos apresentaram soluções que atingiram valores iguais ou muito próximos do ótimo. Os tempos de obtenção das soluções, porém variaram entre 95,75 e 1186 minutos em um microcomputador Apollo Domain 3000.

Métodos de Programação Dinâmica

Um algoritmo para o PPV padrão de Programação Dinâmica, foi proposto inicialmente por [Eilon et al 71]. Considerando um PPV com um número fixo de veículos. Seja $c(S)$ referenciado ao custo ou comprimento de uma rota de um veículo através dos vértices 1 e todos os vértices de um subconjunto S de $V \setminus \{1\}$. Então o custo mínimo pode ser determinado pela seguinte recursão:

$$f_k(U) = \begin{cases} c(U), & k = 1; \\ \min_{U^* \subset U \subseteq V \setminus \{1\}} [f_{k-1}(U \setminus U^*) + c(U^*)], & k > 1 \end{cases} \tag{V.25}$$

O custo da solução é igual a $f_m(V \setminus \{1\})$ e ao custo da solução ótima dos subconjuntos otimizados U^* em V.25.

Vê-se que $f_k(U)$ tem que ser computado para todos os valores de k e para todos os subconjuntos U de $V \setminus \{1\}$. O número de cálculos necessários é muito elevado para a maioria das instâncias. O uso eficiente da programação dinâmica necessita de uma redução substancial no número de estados através de um procedimento de relaxação, ou usando critérios de viabilidade ou de dominância. Por exemplo, no PPV os conjuntos U e U^* devem satisfazer,

$$\sum_{i \in V \setminus \{1\}} q_i - (m-k)Q \leq \sum_{i \in U} q_i \leq kQ \quad (k = 1, \ldots, m) \text{ e}$$
$$\sum_{i \in U} q_i - (k-1)Q \leq \sum_{i \in U^*} q_i \leq Q \quad (k = 1, \ldots, m) \tag{V.26}$$

A relaxação *Estado de Espaços* provê outra eficiente maneira de reduzir o número de estados, [Christofides et al 81b]. Através da determinação de um limite mais longo sobre o custo da solução ótima, o ótimo pode ser atingido por estar embutido no procedimento de limite (enumerativo). De forma resumida o método pode ser apresentado como segue:
Considerando uma recursão geral de Programação Dinâmica,

$$f_{0,i}(0,j) = \min_{k \in \Delta^{-1}(j)} [f_{0,i-1}(0,k) + c_i(k,j)] \tag{V.27}$$

onde, $f_{0,i}(0,j)$ é o menor custo de ir do estado 0 ao estado j pelo estagio i e $\Delta^{-1}(j)$ é o conjunto de todos os possíveis estados j que podem ser atingidos diretamente, e $c_i(k,j)$ é o custo de ir do estado k no estagio $i-1$ ao estado j no estagio i.

Seja $g(.)$ o mapeamento do *estado de espaços* S associado a V.27 ao estado de espaço T de menor cardinalidade, e seja $F^{-1}(g(j))$ um conjunto satisfazendo a,

$$k \in \Delta^{-1}(j) \Rightarrow g(k) \in F^{-1}(g(j)) \tag{V.28}$$

A recursão V.27 então torna-se:

$$f_{0,j}(g(0), g(j)) = \min_{t \in F^{-1}(g(j))} [f_{0,i-1}(g(0), t) + \bar{c}_i(t, g(j))] \tag{V.29}$$

onde,

$$\overline{c}_i(t, g(j)) \;=\; \min[c_i(k,l) : g(k) = t, g(j) = g(l)] \tag{V.30}$$

Resultando em,

$$f_{0,j}(g(0), g(j)) \;\leq\; f_{0,j}(0, i) \tag{V.31}$$

Esta relaxação é importante, se e só se,

1. $F^{-1}(.)$ pode ser facilmente determinada: isto acontecerá se $g(.)$ é separável, assim dado $g(U)$ e r, $g(U \setminus \{r\})$ pode ser calculado;
2. $g(.)$ é tal que a otimização de V.30 está acima de um pequeno domínio, ou que um bom limite inferior sobre $\overline{c}_i(t, g(j))$ pode ser calculado.

[Christofides et al 81b] usaram a seguinte relaxação para os PPVs. Seja $f_k(U, r)$ o menor custo de suprimento de um conjunto U de vértices, usando k veículos, onde os últimos vértices de k correspondendo a rotas pertencentes a $(2, \ldots, r), (k \leq r \leq n)$. Seja $c(U, r)$ o custo da solução do PCV através de $U \cup \{1\}$, onde o último vértice antes do depósito é r. A recursão é dada por:

$$f_k(U, r) = \begin{cases} \min[f_k(U, r-1), min_{u^* \in U}\{f_{k-1}(U \setminus U^*, r-1) + c(U^*, r)\}], & k, r > 1 \\ c(U, r), & (k = 1) \end{cases} \tag{V.32}$$

Sujeito a V.26. Para este problema, a função de mapeamento é dada por:

$$g(U) \;=\; \sum_{i \in U} q_i \tag{V.33}$$

A recursão de V.29 então se torna,

$$f_k(g(U), r) \;=\; \min[f_k(g(U), r-1), \min_p\{f_{k-1}(g(U) - p, r-1) + \overline{c}(p, r)\}] \tag{V.34}$$

sujeito a,

$$g(V) - (m-1)D \;\leq\; p \;\leq\; \min(g(U), D) \tag{V.35}$$

Alguns limites inferiores foram obtidos para instâncias da literatura, em problemas contendo de 10 a 25 vértices. A razão entre o limite inferior (LI) e o valor das soluções ótimas obtidas variaram entre 93,1% e 100%. Anos mais tarde, [Christofides 85] reportou que instâncias com até 50 vértices poderiam ser resolvidas com este método.

Métodos de Programação Inteira

Particionamento de Conjuntos e Geração de Colunas

[Balinski & Quandt 64] foram os primeiros a trabalhar com este tipo de formulação para o PPV. Nela, tem-se o seguinte esquema geral: Considerando J o conjunto formado por todas as rotas viáveis j e a_{ij} é o coeficiente binário igual a 1 se e só se o vértice $i > 1$ aparece na rota j. Seja c^* o custo ótimo da rota j e x_i, uma variável binária igual a 1 se e somente se a rota j é usada na solução ótima.

O problema pode ser formulado como segue:

$$\text{(PPV Partição) Minimizar} \sum_{j \in J)} c_j^* x_j \tag{V.36}$$

Sujeito a:

$$\sum_{j \in J} a_{ij} x_j = 1 \quad (i \in V \setminus \{1\}) \tag{V.37}$$

$$x_j \in \{0,1\} \quad (j \in J) \tag{V.38}$$

Duas dificuldades principais estão associadas a esta formulação:

1. O número de variáveis binárias x_j pode chegar à casa dos milhões na maioria dos casos reais. Somente problemas muito restritos (ou seja, problemas com poucas soluções viáveis), permitirão que o número de variáveis seja pequeno o bastante de modo que seja possível a solução dos problemas;

2. A dificuldade de se calcular os valores c_j^* pode complicar sobremaneira o problema. Uma vez que toda rota j corresponde a um conjunto de vértices S_j satisfazendo a:

$$\sum_{i \in S_j} q_i \leq Q.$$

O valor de S_j é então obtido pela solução de um PPV sobre S_j.

Se o número de variáveis é pequeno e o objetivo é minimizar o número de rotas (veículos), ou seja $c_j^* = 1$ para todo $j \in J$, a relaxação linear do PPV-Partição sempre atinge uma solução inteira, [Laporte 92]. Se a solução (x^*) não é inteira e dá valores fracionários no valor da função objetivo, então o plano de corte

$$\sum_{j \in J} x_j > \lceil \sum_{j \in J} x_j^* \rceil$$

pode ser introduzido. Muito poucos cortes são geralmente necessários de modo a atingir a integralidade, [Orloff 76].

A dificuldade natural do problema é o uso da geração de colunas. Muitos autores, porém, têm explorado esquemas eficientes, tais como: [Orloff 76] [Foster & Ryan 76], [Desrosiers et al 84], [Skitt & Levary 85], [Agarwal et al 89], [Desrochers et al 92], [Hjorring 95].

Na geração de colunas, um problema reduzido contendo apenas um subconjunto restrito de todas as colunas possíveis (variáveis) é resolvido repetidamente. A relaxação linear do problema reduzido provê um vetor ótimo das variáveis duais λ. A verificação da otimalidade implica no cálculo da coluna s de menor custo marginal, ou seja, na determinação da coluna s satisfazendo a:

$$c_s^* - \lambda y_s = \min_{j \in J}\{c_j^* - \lambda y_j\} \qquad \text{(V.39)}$$

onde y_j é o vetor coluna da restrição de coeficientes de variável x_j. Se o custo marginal de x_s é não negativo, a solução atual é ótima e o procedimento termina. Caso contrário, x_s entra na base e o problema é reotimizado. Uma vez que as soluções do PPV devem ser inteiras, este procedimento deve ser usado em conjunto com um procedimento B&B. A resolução de V.39 é feita usando um algoritmo de caminho mínimo com restrições do mesmo modo que no problema original de percurso de veículos.

O sucesso no uso deste método deve-se em geral aos trabalhos realizados com problemas tipicamente muito restritos tais como o PPV com Janelas de Tempo (PPV-JT) para instâncias com até 100 vértices e o PPV com Múltiplos Depósitos (PPV-MD) para instâncias com até 300 vértices, [Desrosiers et al 84], [Desrochers et al 92], [Ribeiro & Soumis 94].

Formulação em Variáveis de Fluxo com Três Índices

[Fisher & Jaikumar 81] desenvolveram uma formulação do PPV em fluxo com três índices, restrições de capacidade dos veículos, janelas de tempo e tempos de serviço fixo por cliente. Tal formulação usa variáveis para representar

a passagem de um veículo em uma ligação ij. Esta formulação é uma extensão do modelo (PPV-G) apresentado no início desta seção. Além da formulação, os autores também desenvolveram um algoritmo baseado no modelo proposto. Apesar do algoritmo parecer ter sido usado apenas para prover uma solução heurística para o problema, ele garante uma solução ótima em um número finito de passos, caso seja executado até o fim. Uma vantagem do modelo é que ele não requer que os veículos sejam idênticos, como no caso do PPV padrão.

Na essência, dois problemas conhecidos formam o modelo (PPV-G),

1. Problema de Atribuição Generalizado (PAG), obtido pela relaxação das restrições de atribuição de demandas a percursos com capacidade limitada, [Martello & Toth 90];
2. As restrições de Janelas de Tempo (PPV-JT).

No algoritmo heurístico proposto Fisher & Jaikumar, os seguintes passos são relevantes:

Algoritmo de Atribuição Generalizada para o PPV

Passo 1: Inicializar um conjunto de clientes *sementes* $i_1, \ldots, i_K$ os quais são atribuidos a K veículos respectivamente.

Passo 2: Chamar d_{ik} como o custo de inserir o cliente i na rota na qual o veículo k percorre indo do depósito ao cliente i_k e retornando. Ou seja,

$$d_{ik} = \min[c_{1i} + c_{ii_k} + c_{i_k 1}, c_{1i_k} + c_{i_k i} + c_{i1}]$$

Passo 4: Tomar a demanda do cliente i como a sua contribuição ao preenchimento do veículo da rota k.

Passo 5: Construir a rede de atribuição (c_i^k, q_i^k) e aplicar um algoritmo GAP. Guardar a solução.

Passo 6: Aplicar uma nova regra de seleção dos clientes, usando regras de custos de inserção modificada propostas por [Gaskell 67] e [Yellow 70], ou um outro critério de seleção. Se não houver mais regras a testar, terminar retornando a melhor solução obtida, caso contrário ir para o passo 2.

Os autores propuseram um algoritmo exato baseado na decomposição de Benders, [Benders 62]. O procedimento interage entre a solução de um procedimento mestre do PAG, que atribui vértices a veículos e resolve o PPV-JT para determinar a melhor rota de cada veículo. O método tem a vantagem de produzir

uma solução viável, mesmo que não seja executado até o final. Também, uma vez resolvendo de forma repetitiva o PAG e um PPV-JT, isto pode beneficiar diretamente a qualquer melhoria nos algoritmos destes dois problemas. Tais melhorias foram propostas por [Martello & Toth 90] e por [Desrochers et al 92]. Fisher & Jaikumar mostraram resultados computacionais para as instâncias testes da literatura (50 a 199 clientes), os quais superaram todos os resultados dos algoritmos até então produzidos, e em tempo comparável ao algoritmo de economias.

Formulação de Fluxo com Dois Índices

A formulação de fluxo com dois índices é um modelo mais compacto que se aplica aos casos com frota homogênea. Esta formulação foi inicialmente usada para o PPV simétrico, e considerada por [Laporte et al 85].

Seja dado $v(S)$ o limite inferior sobre o número de veículos necessários a servir todos os clientes de S. Para esta formulação, x_{ij} indica o número de vezes (0,1 ou 2) que o elo (i,j), $i < j$ é usado na solução ótima. Um elo com $x_{ij} = 2$ representa uma rota com um único cliente. Os x_{ij}'s são definidos apenas se as condições seguintes são satisfeitas:

1. $i < j$ (x_{ij} é interpretado como sendo igual a x_{ji} sempre que $i > j$);
2. $q_i + q_j \leq Q$;
3. $t_{1i} + t_{ij} + t_{j1} + \delta_i + \delta_j \leq T$

$$\text{(PCV-2F) Minimizar} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{V.40}$$

Sujeito a:

$$\sum_{j=2}^{n} x_{ij} = 2m \quad (i = 1, \ldots, n) \tag{V.41}$$

$$\sum_{i<k} x_{ij} + \sum_{j>k} x_{kj} = 2 \quad (i = 2, \ldots, n) \tag{V.42}$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - v(S) \quad (S \subset V \setminus \{1\} : 2 \leq |S| \leq n-2) \tag{V.43}$$

$$x_{1j} \in \{0,1,2\} \quad (j = 2, \ldots, n) \tag{V.44}$$

$$x_{ij} \in \{0,1\} \quad (i,j = 2, \ldots, n) \tag{V.45}$$

As restrições V.41-V.42 representam o grau dos vértices, enquanto que as restrições V.43 proibem rotas inviáveis que não formam subrotas conectadas ao depósito, e não atendem às restrições de capacidade ou de distância/tempo.

É importante notar que m pode ser tomado como uma constante fixa ou como uma variável. Como no caso assimétrico é sempre bom impor limites inferiores e superiores sobre m que pode ser tomado como uma constante fixa ou como uma variável. Na restrição de eliminação de subrotas, $v(S)$ é um limite inferior sobre o número de veículos necessários à visita de S. Como no PCV, a validade destas restrições podem ser estabelecidas da identidade,

$$\sum_{k \in S}\left(\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj}\right) = 2 \sum_{\substack{i,j \in S \\ i<j}} x_{kj} \sum_{\substack{i,j \in S \\ i<j}} x_{ij} + \sum_{\substack{i \in \overline{S}, j \in S \\ \text{ou } i \in S, j \in \overline{S}}} x_{ij}$$

e as restrições de conectividade,

$$\sum_{\substack{i \in \overline{S}, j \in S \\ \text{ou } i \in S, j \in \overline{S}}} x_{ij} \geq 2v(S) \; (S \subset V \setminus \{1\}; 3 \leq |S| \leq n-3)$$

Estas restrições forçam que pelo menos $2v(S)$ existam entre S e seu complemento $\overline{S}$, se $v(S)$ veículos são usados para S. Assim como no caso assimétrico, o valor de $v(S)$ pode ser tomado como um limite do problema de *binpacking* para o PPV capacitado.

O seguinte algoritmo de relaxação poderá resolver o modelo proposto.

Passo 1: (Inicialização) Definir um primeiro subproblema contendo (função objetivo, V.40), o grau das restrições V.41 e V.42, tanto quanto os limites sobre as variáveis. Inserir este subproblema numa fila. Seja z^* o valor da melhor solução conhecida do PPV (Se nenhuma solução é conhecida, fazer $z^* = \infty$).

Passo 2: (Checar Finalização) Se a fila está vazia, finalizar. caso contrário, extrair dela o próximo subproblema a ser resolvido.

Passo 3: (Solução do Problema) Resolver o subproblema (um PPL). Seja $\overline{z}$ o valor de sua solução.

Passo 4: (Verificação de Viabilidade) Se $\overline{z} \leq z^*$ provar a otimalidade do subproblema atual e ir ao passo 2. Caso contrário, verificar se qualquer violação das restrições de eliminação de subrotas podem ser identificadas. Se pode, gerar uma ou muitas destas restrições e ir ao passo 3. Se nenhuma violação das

restrições de subrotas puder ser detectada, verificar se a solução é inteira. Se ela é, fazer $z^* = \overline{z}$ e ir ao passo 2. Se a solução não é inteira, criar subproblemas pela ramificação sobre as variáveis fracionárias; inserir estes subproblemas na fila e ir ao passo 2.

Este algoritmo foi proposto por [Laporte et al 85] tendo sido aplicado a instâncias de 10 a 50 vértices em tempos que variaram de 2,19 a 109 segundos para instâncias euclidianas, e de 1,69 a 107 segundos para as não euclidianas, em um computador CYBER 173 usando um compilador FTN5. O algoritmo mostrou-se robusto para as instâncias euclidianas até 50 vértices, enquanto que para as não euclidianas obteve um funcionamento aceitável em instâncias até 60 vértices.

V.1.2 Principais Heurísticas do PPV Genérico

Os algoritmos heurísticos para o PPV podem ser derivados de procedimentos do PCV. Os algoritmos do vizinho mais próximo, de inserção e de melhorias de rotas podem ser aplicados ao PPV quase sem nenhuma modificação. Entretanto, quando aplicados ao PPV deve-se tomar um certo cuidado na formação de rotas sempre viáveis. O algoritmo de [Fisher & Jaikumar 81], descrito acima, é heurístico pois provê uma solução viável do PPV a cada passo, sendo interrompido em geral antes que a otimalidade seja atingida.

Descreve-se a seguir algumas das heurísticas mais importantes desenvolvidas para o PPV, as quais vêm sendo largamente utilizadas. Também, mostra-se um pouco do desenvolvimento atual em procedimentos de busca local metaheurísticos.

Algoritmo de Economias

Passo 1: (Cálculo de Economias) Supondo $s_{ij} = c_{i1} + c_{1j} - c_{ij}$, $\forall\, i,j = 2,\ldots,n$, criar $n-1$ rotas de veículos $(1,i,1)$, $(i = 2,\ldots,n)$. $O(n^2)$

Passo 2: (Ordenar Economias) Ordenar as economias s_{ij} em ordem descendente. $O(n^2 log\ n)$

Passo 3: (Juntar Rotas) Considerar dois veículos contendo os arcos $(i,1)$ e $(1,j)$ respectivamente. Se $s_{ij} > 0$, tentar juntar estas rotas pela introdução do arco (i,j) e pela deleção do arco $(i,1)$ e $(1,j)$, figura V.3. Implementar a ligação se isto resulta numa rota viável. Repetir este passo até que nenhuma melhoria seja possível. $O(n^2)$

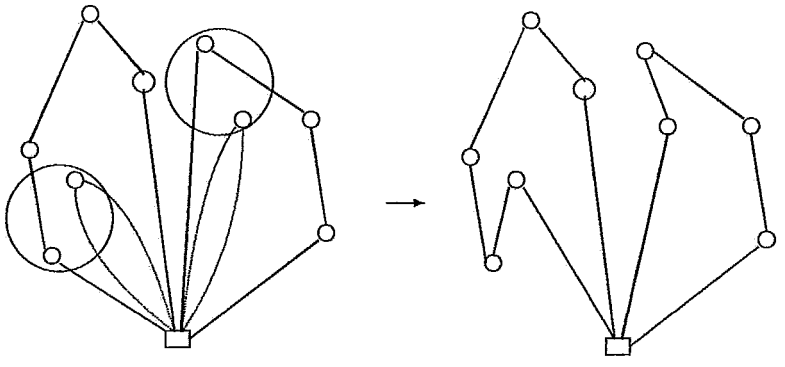


Figura V.3: Juntando clientes a rotas usando economias.

Passo 4: (Reotimizar Rotas) Reotimizar rotas usando algum procedimento do PCV. Terminar. $O(n^k)$

O algoritmo original pode ser modificado através de um número de recursos que são em geral explorados nas etapas de geração de economias, onde está concentrada toda a complexidade do algoritmo. Algumas técnicas de melhorias foram propostas com o objetivo de obter uma melhoria nas soluções do método, [Gaskell 67], [Yellow 70], [Mole & Jameson 76], [Golden et al 77], [Nelson et al 85], [Altinkemer & Gavish 91], [Paessens 88] e [Negreiros 93]

O algoritmo de economias como proposto acima, ignora implicitamente os custos fixos dos veículos os quais podem ser facilmente levados em conta pela adição desta constante a todo c_{1j} $(j = 2, \ldots, n)$. Soluções com um número fixo de veículos podem ser obtidas pela repetição do passo 3 até que o número de rotas seja atingido, mesmo que as *economias* geradas sejam negativas.

[Negreiros 93] coloca uma proposta de melhoria do algoritmo de economias, utilizando a observação de que os resultados obtidos são pobres em virtude do processo de junção de duas rotas só se dar pelos extremos da rotas já formadas. Este fato faz com que o algoritmo descarte boas *economias* que podem posicionar novos elementos no centro de uma rota já formada, permitindo com isto resultados bem melhores, figura V.4.

Um outro procedimento que pode ser usado como uma etapa final, ao invés do passo 4, seria a realização de trocas locais entre rotas, considerando no problema das trocas a manutenção da viabilidade. A vantagem principal desta pós-otimização é o aproveitamento da solução obtida na busca de melhores resultados, incluindo o refinamento das rotas geradas por um procedimento do PCV de trocas locais. Esta idéia foi proposta por [Russel 77], utilizando o procedimento 3-Otimal

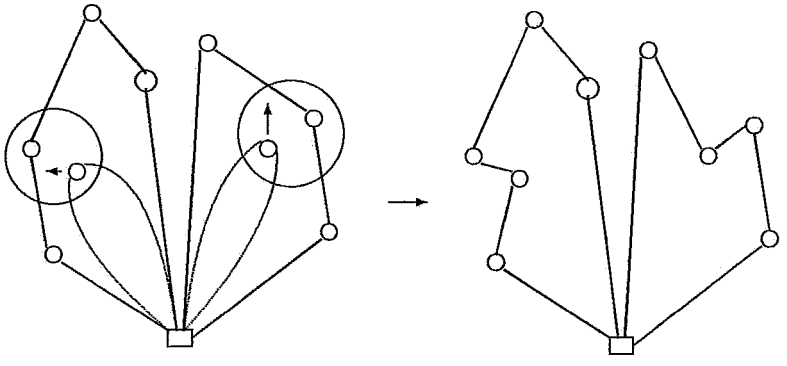


Figura V.4: Proposta de Negreiros para melhoria do Algoritmo de Economias.

de Lin.

Algoritmo Paralelo de Economias com Inserção

Passo 1: (Cálculo de Economias) Supondo $s_{ij} = c_{i1} + c_{1j} - \alpha c_{ij}$, $\forall\, i, j = 2, \ldots, n$, que satisfaça as restrições consideradas.

Passo 2: (Ordenar Economias) Ordenar as economias s_{ij} em ordem descendente.

Passo 3: (Juntar Rotas) Considerar a economia s_{ij} corrente. Se i e j não foram atendidos, abrir uma nova rota. Caso contrário, se i está numa rota e j não foi atendido, incluir j na *melhor* posição de inserção na rota de i - antes de i ou depois, desde que a entrada de j não viole as restrições consideradas. (Aplicar o mesmo raciocínio para j atendido e i não atendido), figura V.4. Ler uma nova economia s_{ij} e repetir este passo até que todas economias tenham sido exploradas.

Passo 4: (Finalizar com clientes não atendidos) Seja d_{1i1} o custo de atendimento de um cliente não atendido no passo 2, abrir novas rotas (individuais) apenas para aqueles onde as restrições não são violadas.

Passo 5: (Reotimizar Rotas) Reotimizar rotas usando algum procedimento do PCV. Terminar.

Passo 5': (Troca entre rotas - Alternativa) Utilizar um procedimento que permita realizar uma troca entre clientes de duas rotas vizinhas, mantendo o atendimento das restrições do problema. Fazer isto para todas as rotas encontradas nos passos 4 e 5, e reotimizar rotas individualmente com um procedimento do PCV.

Nos testes computacionais realizados, foi considerado o desempenho da melhor solução encontrada pelo procedimento de melhoria do algoritmo de Clarke & Write, onde se manteve o parâmetro α proposto por [Yellow 70], para o formato da rota, e o parâmetro de *grade*, proposto por [Golden et al 77], que procura fundamentalemente reduzir o número de economias a serem calculadas, em função de valores factíveis de ligação entre duas demandas consideradas.

A seguinte tabela reflete os resultados obtidos com a melhoria aqui proposta, a partir dos problemas clássicos da literatura, [Eilon et al 71]. Admitiu-se que:

$$c_{ij} = \lfloor \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \rfloor \tag{V.46}$$

onde, (x_i, y_i) são as coordenadas da demanda ou depósito i.

Tabela V.1: Resultados obtidos com a proposta de melhoria.

Prob	# N	Qv	Tmáx	Economia+Inserção					Literatura	
				α	*grade*	TP	NR	Ctot	NR	Ctot
1	7	3	-	0.9	1	5.00	2	119	2	114
2	13	6000	-	1	0.5	16.15	4	290	4	290
3	22	6000	200	1	0.01	9.50	4	591	4	585
4	23	4500	240	1	0.01	9.39	5	955	5	949
5	30	4500	240	0.85	0.03	9.44	5	951	4	875
7	33	8000	240	1	0.03	15.55	4	803	4	810
8	51	160	-	1	0.04	28.67	5	539	5	524
9	76	149	-	0.85	0.15	1:47.59	10	857	10	857
10	101	200	-	0.87	0.02	3:35.63	8	857	8	833
11	151	200	-	0.95	0.009	4:59.12	12	1089	12	1014
12J	200	200	-	0.90	0.1	5:08.24	17	1358	17	1386
8T	51	160	200	1	0.075	16.36	6	597	6	560
9T	76	140	160	0.85	0.027	1:24.21	12	905	12	916
10T	101	200	230	0.90	0.01	3:13.94	9	897	9	885
11T	151	200	200	0.87	0.67	2:15.99	15	1292	15	1217
12TJ	200	200	200	0.90	0.07	3:54.26	19	1475	19	1508

- #N : Número de vértices mais depósito;
- Qv : Capacidade do veículo;
- Tmáx : Máximo tempo de serviço por rota;
- α : Parâmetro de formato da rota;

- *grade* : Parâmetro de junção de duas demandas;
- TP : Tempo de processamento (mm:ss.ds)[1];
- NR : Número de rotas geradas;
- Ctot : Custo Total.

A menos do problema 5, todas os testes tiveram o mesmo número de rotas, e os custos totais entre o algoritmo que obteve o melhor resultado considerado pela literatura e o Algoritmo de Economia+Inserção estão sempre muito próximos, sendo que em algumas situações este algoritmo supera os melhores resultados até então publicados. A maior vantagem obtida deste método está na comparação *desempenho x resultado obtido.* Não foram consideradas as melhorias propostas no *Passo 5'*, uma vez que esta última vem das idéias retiradas dos procedimentos metaheurísticos.

Algoritmo de Varreduras - *Sweep*

Apesar de as origens do método estarem relacionadas ao trabalho de [Wren 71], o método é atribuido a [Gillett & Miller 74], pois foi quem deu o seu nome. De modo a facilitar a implementação deste método, os autores propuseram o uso da representação dos vértices por suas coordenadas polares (θ_i, r_i), onde θ_i é o ângulo e r_i é o comprimento do raio. Atribui-se o valor de $\theta_i = 0$ a um vértice arbitrário i^* (podendo ser o depósito em caso de existir apenas um) e calcular os ângulos remanescentes a partir de $(1, i^*)$. Ordena-se os vértices na ordem ascendente dos seus ângulos.

Passo 1: Escolher um veículo não usado k.

Passo 2: Iniciando de um vértice não usado tendo o menor ângulo, atribuir vértices ao veículo tantos quantos não excedam a capacidade do veículo e/ou suas restrições de tempo/distância. Se vértices não atribuidos a rotas ainda permanecem, ir ao passo 1.

Passo 3: Otimizar cada rota de veículo separadamente pela resolução de um método do PCV correspondente. Realizar trocas de vértices entre rotas adjacentes se isto melhora a distância. Reotimizar e parar.

O método *sweep* pode ser caracterizado basicamente pelos passos 1 e 2, onde no passo 2 tanto se pode iniciar pelo menor ângulo (*forward sweep*) quanto

[1]Para um computador 386/SX-16MHz sem co-processador.

pelo maior ângulo (*backward sweep*). O passo 3, porém, é um recurso importante na melhoria das rotas, podendo ser feito em tempo razoável através de um procedimento otimal. Este algoritmo, se bem projetado, poderá ser muito eficaz, porém poderá não ter um desempenho razoável em redes simétricas e mistas (não euclidianas).

[Foster & Ryan 76] propuseram uma melhoria ao algoritmo de varredura. Eles ordenaram os clientes, como no método *sweep*, de forma radial porém elaboraram um método mais sofisticado de particionamento dos clientes. No procedimento, todo subconjunto contíguo da ordem radial é considerado como uma rota possível. Destas rotas, aquelas que são viáveis tornam-se um *conjunto de pétalas*, assim chamado devido às rotas serem radiais a partir do depósito como pétalas de uma flor. Através desta aproximação pode-se notar que qualquer solução do algoritmo de varreduras está contido no conjunto de pétalas. Um subconjunto de rotas de um conjunto petal que contém cada cliente exatamente uma vez será uma solução viável para o PPV. Foster & Ryan encontraram o subconjunto ótimo pela formulação do problema como um problema de particionamento. Isto pode ser facilmente resolvido desde que a matriz de restrições seja unimodular sobre certas condições.

Algoritmo de Duas Fases

Este algoritmo foi basicamente projetado para os casos capacitado e restrito no tempo/distância. Ele produz duas soluções alternativas para um dado parâmetro $\lambda \geq 1$ e $\mu \geq 1$ os quais são definidos pelo usuário. A melhor das soluções pode então ser selecionada. Pode-se repetir este procedimento para muitos valores de λ e μ, [Christofides, Mingozzi & Toth 79].

FASE 1 - Construção Sequencial de Rotas

Passo 1: Fazer o índice da primeira rota igual a 1.

Passo 2: Selecionar qualquer vértice não roteirizado i_k para inicializar a rota k. Para cada vértice não roteirizado i, calcular $\delta_i = c_{1i} + \lambda c_{ii_k}$.

Passo 3: Seja $\delta_{i^*} = \min_{i \in S_k}\{\delta_i\}$, onde S_k é o conjunto formado pelos vértices não roteirizados que podem ser inseridos de forma viável na rota k. Inserir o vértice i^* dentro da rota k. Otimizar a rota k usando um algoritmo do PCV. Repetir o passo 3 até que nenhum outro vértice seja atribuido à rota k.

Passo 4: Se todos os vértices foram inseridos nas rotas, Parar. Caso contrário faça $k = k + 1$, ir ao passo 2.

FASE 2 - Construção de Rotas Paralelas

Passo 5: Inicializar k rotas $R_i = (1, i, 1)/(i = 1, \ldots, k)$, onde k é o número de rotas obtidas ao final da fase 1. Seja $K = \{R_1, \ldots, R_k\}$.

Passo 6: Para cada rota $R_i \in K$ e para cada vértice i não associado com uma rota, calcular $\epsilon_{t_i} = c_{1i} + \mu c_{1k_t} - c_{1i_t}$ e $\epsilon_{t^*i} = \min\{\epsilon_{ti}\}$. Associar o vértice i com a rota R_{t^*} e repetir o passo 6 até que todos os vértices tenham sido associados a uma rota.

Passo 7: Tomar qualquer rota $R_t \in K$ e fazer $K = K \setminus \{R_t\}$. Para cada vértice i associado com a rota R_i, calcular $\epsilon_{t_i'} = \min_{R_i \in K}\{\epsilon_{t_i}\}$ e $\tau_i = \epsilon_{t_i'} - \epsilon_{t_i}$

Passo 8: Inserir dentro da rota R_t e o vértice i^* satisfazendo a $\tau_{i^*} = \max_{i \in S_k}\{\tau_i\}$, onde S_t é o conjunto de vértices não roteados associados à rota R_t que pode ser viabilizada quando inserida na rota R_t. Otimizar a rota R_t usando um algoritmo otimal do PCV. Repetir o passo 8 até que não haja mais vértices a serem inseridos na rota R_t.

Passo 9: Se $K = \emptyset$, ir ao passo 6. Caso contrário, se todos os vértices estão roteirizados, Parar. Caso permaneçam vértices não roteirizados, criar novas rotas iniciando pelo passo 1 da fase 1.

A complexidade assintótica deste algoritmo é um pouco difícil de ser calculada, uma vez que existem chamadas aos métodos otimais para cada rota sequencial gerada na fase 1. Sua vantagem reside principalmente na fase 2, com a possibilidade de incorporação de rotas e troca de vértices entre rotas.

Algoritmos de Busca Local Metaheurísticos

No início dos anos 90, uma nova corrente de métodos surgiram na literatura, provocando uma corrida de melhores resultados sobre a série de problemas padrões para testes. Considerados como métodos de melhorias, este procedimentos derivam de metaheurísticas tais como: Busca Tabu, *Simulated Annealing*, Algoritmos Genéticos e Redes Neurais. A seguir discute-se duas técnicas de busca: Busca Tabu e *Simulated Annealing* as quais vêem atingindo grande sucesso na obtenção de soluções do PRV.

1. Busca TABU

   Algoritmos heurísticos baseados em Busca TABU, [Glover 89], [Glover 90b], foram explorados por: [Willard 89], [Pureza & França 91], [Osman 93], [Taillard 93], [Gendreau et al 94], [Hjorring 95], [Xu & Kelly 96] [Rego 96] [Rego & Roucariol 96].

Dentre as técnicas usadas destacam-se os procedimentos GENIUS para o PCV, [Gendreau et al 94], o método de decomposição de pétalas, [Hjorring 95], método de Cadeias de Ejeção, [Rego & Roucariol 96], e o método de Busca Tabu baseado em fluxo em redes, [Xu & Kelly 96].

Uma idéia geral de funcionamento de um procedimento de Busca TABU foi proposta por [Gendreau et al 94]. Nela constrói-se uma sequência de soluções, e então executa-se um passo de melhoria. As sucessivas rotas produzidas para os veículos fornecidas pelo algoritmo não precisam ser viáveis; seu grau de distância da viabilidade é medido através de uma penalidade imposta à função objetivo. Os passos gerais do procedimento são colocados a seguir.

Passo 1: Construir rotas no sentido horário e anti-horário, passos 1 e 2 do algoritmo de varredura. Seja x esta solução, tomar o seu custo como sendo $F(x)$. Fazer a lista TABU $T = \emptyset$.

Passo 2: Definir $N(x)$ a vizinhança de x, como o conjunto de todas as soluções que podem ser atingidas pela inserção de um vértice arbitrário na p-ésima vizinhança usando o procedimento GENI. Se $N(x) \setminus T = \emptyset$, ir ao passo 3. Caso contrário, identificar a solução de menor custo y em $N(x) \setminus T$ e fazer $x = y$. Atualizar a melhor solução conhecida, [Gendreau et al 90].

Passo 3: Se o número máximo de iterações desde o início do processo ou desde a última atualização foi atingido, ir ao passo 4. Senão atualizar T e ir ao passo 2.

Passo 4: Tentar melhorar cada uma das rotas através do critério US de pós-otimização, [Gendreau et al 90].

Este algoritmo foi aplicado ao conjunto de instâncias padrões da literatura, porém mostrou-se bastante caro na obtenção dos novos resultados, os quais foram calculados em uma estação Silicon Graphics 36MHz/5,7 MFlops.

Resultados melhores foram obtidos mais recentemente por [Xu & Kelly 96], assim como em tempos mais razoáveis em uma estação DEC ALPHA. Nos resultados apresentados, a grande relevância é a de que sempre igualaram ou ultrapassaram aqueles produzidos pelo algoritmo de [Fisher 94].

2. <u>*SIMULATED ANNEALING - SA*</u>

[Osman 93] estudou a aplicação de Busca Tabu e SA para o PPV. Em ambos os métodos foram usados os mesmos movimentos básicos, mecanismo de 1-troca entre rotas para modificar a solução atual.

Nos esquemas clássicos de SA, escolhe-se aleatoriamente um movimento da lista de possíveis movimentos os quais podem levar a regiões inexploradas das instâncias, resultando num aumento no tempo computacional de forma não proibitiva. [Osman 93] tomou um método mais esquematizado, onde os

índices das rotas são randomicamente misturados de modo a formar uma permutação Π dos índices $\{1, \ldots, K\}$. O método verifica o emparelhamento de rotas (R_i, R_j) na seguinte ordem:

$$(R_{\pi(1)}, R_{\pi(2)}), \ldots, (R_{\pi(1)}, R_{\pi(k)}), (R_{\pi(2)}, R_{\pi(3)}), \ldots, (R_{\pi(k-1)}, R_{\pi(k)})$$

e para cada par considera todas as 1-trocas de forma simétrica. Quando os emparelhamentos são concluídos, os índices são novamente misturados e a busca continua.

Osman testou seu método sobre o conjunto de testes padrões, e ambos os métodos produziram bons resultados, de melhor qualidade do que aqueles apresentados pelas heurísticas até o início dos anos 90. Entretanto um tempo computacional bem maior foi necessário. Em geral a Busca Tabu apresentou melhores resultados que o SA, que mostrou uma grande variância com respeito à qualidade da solução e o tempo para obtê-la.

O esquema padrão do SA tem um conjunto de parâmetros os quais devem ser especificados de modo a definir o planejamento do *Annealing*. Para a obtenção dos melhores resultados, estes parâmetros devem variar conforme a classe de problema a ser analisado. Infelizmente, porém, a melhor escolha dos parâmetros para uma classe particular de problemas não é obvia.

[Hiquebran et al 94] introduziram um método revisado SA que possui parâmetros que são mais fáceis de entender do que aqueles do procedimento padrão SA. Com o objetivo de aplicar o esquema revisado ao PPV, dois tipos de movimentos são usados na mudança da solução: Trocar e Inserir. Ao invés de selecionar randomicamente e executar uma permuta de clientes entre elas, [Hiquebran et al 94] consideram uma busca mais extensiva. Um cliente, chamado de *cliente alvo*, é aleatoriamente selecionado. Seja a rota do cliente alvo chamada de *rota alvo*, então todas as rotas senão a *rota alvo* serão consideradas para o movimento de troca com o *cliente alvo*. Se um movimento de troca está em teste, um cliente de uma rota não alvo é aleatoriamente testado. O movimento com a maior melhoria sobre todo o par de rotas é escolhido, e então testado usando o critário normal de aceitação do SA.

Os autores apresentaram resultados para oito problemas testes propostos por [Christofides, Mingozzi & Toth 79]. Para cada problema, quatro planejamentos diferentes foram testados e 100 execuções foram realizadas para cada plano. O tempo de cada execução gira em torno de 16 segundos, para um problema com 50 clientes, e até 170 segundos para 199 clientes em uma SUN SPARC 2. As melhores soluções encontradas pelo método revisado SA são de qualidade similar às obtidas por [Osman 93].

V.2 O PPV com Múltiplos Depósitos

O Problema de Percurso de Veículos com Múltiplos Depósitos (PPV-MD), considera que há mais de um local onde os veículos estão baseados, e o objetivo é a geração de percursos da frota que permita inclusive a definição das áreas de influência de um depósito sobre um subconjunto de clientes, figura V.5. A literatura, neste caso, não tem dado tanto destaque a este problema, apesar de nos últimos anos terem surgidos resultados expressivos tanto usando heurísticas quanto métodos exatos, [Wren & Holliday 72], [Gillet & Johnson 76], [Golden et al 77], [Carpaneto et al 89], [Laporte et al 84], [Chao et al 93], [Ribeiro & Soumis 94].

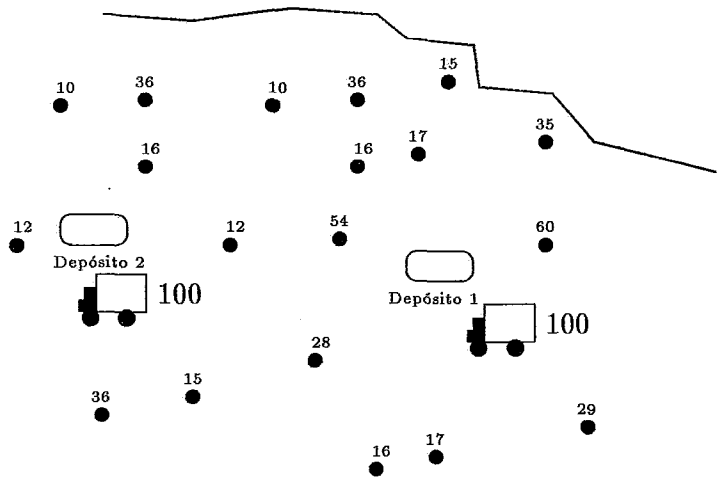


Figura V.5: Situação geral de um PPV-MD

Uma formulação geral do problema proposta por [Carpaneto et al 89], pode ser colocada a partir do modelo de um grafo como segue.

Considerando-se um grafo $G = (V, A)$ definido como, $V = \{1, \ldots, p\}$ (onde $p = n + \sum_{k=1}^{m} r_k$) é o conjunto de vértices particionados em $m+1$ subconjuntos: sendo o subconjunto $N = \{1, \ldots, n\}$, no qual a cada vértice j está associado uma viagem T_j, e os subconjuntos $Q_1, \ldots, Q_m$ onde para $k = 1, \ldots, m$, $Q_k = \{n + \sum_{h=1}^{k-1} r_h\}$ contém r_k vértices associados aos r_k veículos baseados no depósito D_k. $A = \{(i,j) : i, j \in V\}$ é um conjunto de arcos. Para cada arco $(i,j) \in A$, a função é dada por:

$$\overline{c}_{ij} = \begin{cases} c_{ij}, & \text{Se } i,j \in N \\ c_{kj}, & \text{Se } i \in Q_k, j \in N \\ c_{ik}, & \text{Se } i \in N, j \in Q_k \\ 0, & \text{Se } i,j \notin N, i = j \\ \infty, & \text{Se } i,j \notin N, i \neq j \end{cases}$$

O PPV-MD pode ser colocado como: O problema de encontrar o conjunto de circuitos (subrotas), cada uma contendo somente um vértice em $V \setminus N$, tal que todos os vértices em V são visitados exatamente uma vez e a soma dos custos dos arcos na solução é minimizada.

Cada subrota contendo um custo finito e que visita sequencialmente os vértices $v_1, v_2, \ldots, v_h, v_1$ com $v_1 \in Q_k$ (para algum k e $v_2, \ldots, v_h \in N$), corresponde à *tarefa* do veículo sediado no depósito D_k na cobertura das viagens (rotas) $T_{v_2}, \ldots, T_{v_h}$, e seu custo é o custo desta tarefa. As subrotas contendo apenas um vértice v_1 (com $v_1 \in N$) têm custo zero e correspondem a veículos não utilizados. O custo global das subrotas na solução ótima é, pois, igual ao custo total do PPV-MD (Caso o custo global seja infinito, nenhuma solução existe para o PPV-MD).

Nota-se que nenhuma subrota de custo infinito existe no subgrafo induzido pelos vértices de N, portanto qualquer subrota deve visitar pelo menos um vértice em $V \setminus N$. Para isso é necessário que apenas um vértice de $V \setminus N$ seja visitado por cada subrota viável, portanto evitando veículos que partem e terminam suas tarefas em depósitos distintos.

Um modelo de grafos pode ser formulado como um problema de programação inteira, como a seguir.

Sejam dados,

$$x_{ij} = \begin{cases} 1, & \text{Se o arco } < i,j > \text{ é usado na solução ótima} \\ 0, & \text{Caso contrário.} \end{cases}$$

Π, é o conjunto formado por todos os caminhos elementares $\mathcal{P}$ que ligam diferentes vértices em $V \setminus N$.

$$\text{(PPV-MD) Minimizar} \sum_{i,j \in V} c_{ij} x_{ij} \tag{V.47}$$

Sujeito a:

$$\sum_{i \in V} x_{ij} = 1 \quad (j \in V) \tag{V.48}$$

$$\sum_{j \in V} x_{ij} = 1 \quad (j \in V) \tag{V.49}$$

$$\sum_{ij \in \mathcal{P}} x_{ij} \leq |\mathcal{P}| - 1 \tag{V.50}$$

$$x_{ij} \in \{0,1\} \quad (i,j \in V) \tag{V.51}$$

As restrições V.48-V.49 são as restrições de atribuição, permitindo que cada solução viável corresponda à família de subrotas disjuntas que cobre todos

os vértices. As restrições V.50 permitem que cada subrota visite apenas um vértice em $V \setminus N$, desde que no caminho $|\mathcal{P}|$ ligando vértices diferentes em $V \setminus N$ seja permitido.

É importante notar que a família II de caminhos proibidos pode ser restrita a conter apenas caminhos elementares ligando vértices v_1 e v_2 com $v_1 \in Q_{k_1}$, $v_2 \in Q_{k_2}$ e $k_1 \neq k_2$, que são caminhos os quais juntam vértices correspondentes a veículos baseados em diferentes depósitos. Desta forma, subrotas visitando diferentes vértices v_1 e v_2 com $v_1, v_2 \in Q_k$ são viáveis. Entretanto, é sempre possível obter uma solução equivalente na qual a subrota visita o vértice v_1 e não visita o vértice v_2. Neste caso, seja $(v_1, i_1, \ldots, i_h, v_2, j_1, \ldots, j_l, v_1)$ uma sequência de vértices visitados na subrota, tal subrota pode ser trocada, sem nenhum custo adicional, em duas subrotas $(v_1, i_1, \ldots, i_h, v_1)$ e $(v_2, j_1, \ldots, j_l, v_2)$.

[Carpaneto et al 89] aplicaram um procedimento B&B associado a um procedimento de Limite Aditivo (encontra limite inferior). As instâncias consideraram de 10 a no máximo 70 clientes, tendo analisado também a sensibilidade da presença de dois e três depósitos em um computador HP9000/840s. Uma importante nota colocada, foi sobre a dificuldade na obtenção das soluções a partir dos três depósitos. O resultado obtido com a solução inicial foi no entanto muito expressivo, ou seja, o algoritmo proposto atingiu em média valores inferiores a 1% de distância da solução de Limite Inferior.

[Ribeiro & Soumis 94] propuseram uma nova aproximação para o problema, baseada na solução da relaxação contínua do problema através de um esquema de geração de colunas. Eles mostraram que um limite dado pela relaxação linear do fluxo inteiro da formulação de multi-comodidade, é tão bom quanto o proposto por [Carpaneto et al 89], que considera a atribuição de relaxação de caminhos mínimos. Uma nova formulação para o problema tomando o Problema de Particionamento de Conjuntos com Restrições Laterais, assim como um esquema de geração de colunas para a solução exata de sua relaxação contínua foi proposto.

[Ribeiro & Soumis 94] mostram que o limite obtido pela geração de colunas é igual àquele dado pela relaxação linear do problema inteiro da formulação multi-comodidade, portanto é melhor que o limite aditivo de [Carpaneto et al 89]. Resultados computacionais são reportados para tempos computacionais que giram em torno de 100 a 3300 segundos em instâncias com 100 a 300 clientes e de 2 a 6 depósitos.

No procedimento heurístico proposto por [Gillet & Johnson 76], aproveita-se as idéias do algoritmo de varredura para realizar um particionamento dos clientes em grupos que são atribuidos a depósitos individuais. Eles consideram que o padrão de agrupamento é uma função da distribuição espacial dos clientes e das restrições de carga e da distância ou tempo.

Cada grupo é composto de uma simples cadeia de pontos de demanda os quais circundam um único depósito. A intenção neste caso não é a de formar rotas viáveis com estas cadeias, é de fato uma necessidade de representação do arranjo dos pontos na periferia dos grupos. Os grupos são construidos pela inserção de pontos não atribuidos, um a um, dentro dos grupos parcialmente formados. O algoritmo tenta pois considerar a distância de separação dos grupos minimizando, para cada cliente, a distância extra que um veículo deveria percorrer para incluir aquele cliente j na cadeia entre os pontos i e k, que é calculada da expressão $d_{ij} + d_{jk} - d_{ik}$, onde d é a distância simétrica entre pontos da rede, (depósito + clientes).

Os clientes são atribuidos aos depósitos, de tal forma que grupos iniciais são formados inicialmente muito próximos aos depósitos e se expandem gradualmente para longe. O critério de inserção de mínimo custo é o critério para a determinação do depósito a cada cliente. Para résolver o problema de atribuição na região equidistante a dois clientes, um segundo critério é empregado. Se j é o ponto de demanda a ser considerado para atribuição, o algoritmo determina o cliente k não atribuido que é mais próximo a ele. Uma estimativa de custo de atribuir j e k a cada um dos dois grupos próximos deve ser feita. Se o depósito com o mínimo custo de inserção para apenas j também tem o menor custo estimado de inserção para os dois pontos j e k, então j é atribuido àquele depósito. Caso contrário, a atribuição de j é descartada até que o ponto k tenha sido considerado para atribuição. Durante este processo de formação de grupos a restrição de distância ou tempo deve ser verificada de modo que nenhum cliente seja atribuido a um depósito muito distante.

A solução inicial obtida poderá ser melhorada com trocas locais entre rotas de clientes que estão situados nas regiões entre depósitos. Cada depósito é tomado como um *centro atrator* e tenta drenar estes clientes dos depósitos adjacentes. O cliente candidato a reatribuição, assim como alguns de seus vizinhos no mesmo grupo, são declarados não atribuidos e o processo de formação dos grupos é reiniciado. Se a localização de um candidato é reatribuida a um grupo completamente formado ligado ao depósito atrator, o procedimento de atribuição continua até o final e dois novos problemas de um único depósito é resolvido novamente. Se uma solução melhorada acontece, as mudanças tornam-se parte da melhor solução, e esta torna-se a base para os futuros refinamentos.

Um fato relevante na heurística de [Gillet & Johnson 76], foi notado que à medida que o número de terminais cresce, uma redução real no tempo de processamento acontece, em virtude das novas possibilidades de reatribuição de carga.

V.3 O PPV com Janelas de Tempo

O Problema de Percursos com Janelas de Tempo (PPV-JT) considera que pedidos de certos clientes devem ser atendidos dentro de uma ou várias (múltiplas janelas) faixas de horário, em virtude da necessidade de organização dos recebimentos ou entregas de produto em grandes lojas ou supermercados, na priorização do atendimento a clientes especiais, ou devido a influências nos horários de tráfego que dificultam o atendimento de clientes em determinadas zonas críticas. Em geral o PPV-JT está relacionado a aplicações em tempo real (roteamento dinâmico), onde através de contatos telefônicos os clientes são atendidos e imediatamente atribuidos ao serviço da frota disponível.

A literatura tem desenvolvido bastante neste assunto, vários trabalhos foram feitos no encaminhamento de soluções as quais voltaram-se aos problemas especiais básicos do PCV e do m-PCV com janelas de tempo, [Backer & Schaffer 86], [Christofides et al 81b], [Desrosiers et al 88]; ao problema de serviço por chamada telefônica (*Dial-a-Ride*), [Psaraftis 83], [Sexton & Bodin 85a], [Jaw et al 86], [Sexton & Bodin 85b]; no problema de escalonamento de ônibus escolares, [Orloff 76], [Swersey & Bellard 84], [Desrosiers et al 84]; no roteamento e aprazamento de operações de navios, [Ochi 89], [Ochi & Maculan 89], [Carvalho & Ochi 92]; na distribuição de medicamentos em hospitais, [Michelon et al 94], dentre outras.

Um modelo matemático baseado em programação inteira foi proposto por [Koskosides et al 92], como segue.

Sejam dados:

$\mathcal{C}$ um conjunto de clientes ($i = 1, \ldots, I$);

$\mathcal{H}$ um conjunto de veículos ($k = 1, \ldots, K$), onde o índice k também significa a rota realizada pelo veículo k;

$\mathcal{N}_k$ é o conjunto de clientes i servidos pelo veículo k, denominado $\mathcal{N}_k = \{i : y_{ik} = 1\}$ onde $I \equiv \bigcup_k \mathcal{N}_k$;

c_{ij}^k o custo de viagem diretamente do cliente i ao cliente j, com o veículo k;

q_i é o tamanho do pedido do cliente i;

Q_k é a capacidade do veículo k;

t_{ij} é o tempo de viagem entre os clientes i e j;

s_i é o tempo de servir (carga ou descarga) o cliente i;

a_i é o início da janela de tempo do cliente i e b_i é o fim;

T é uma constante maior do que o tempo total de viagem para qualquer rota viável.

Por fim, sejam as seguintes variáveis:

$$x_{ij}^k = \begin{cases} 1, & \text{Se o veículo } k \text{ vai do cliente } i \text{ ao } j \text{ diretamente} \\ 0, & \text{Caso contrário.} \end{cases}$$

$$y_i^k = \begin{cases} 1, & \text{Se o veículo } k \text{ serve o cliente } i \\ 0, & \text{Caso contrário.} \end{cases}$$

t_i é o momento de chegada no cliente i.

$$\text{(PPV-JT) Minimizar} \sum_k \sum_i \sum_j c_{ij}^k x_{ij}^k \tag{V.52}$$

Sujeito a:

$$\sum_{i \in \mathcal{C}} q_i y_i^k \leq Q_k \quad (k \in \mathcal{H}) \tag{V.53}$$

$$\sum_{k \in \mathcal{H}} y_i^k = \begin{cases} K, & i = 0 \\ 1, & i = 1, \ldots, I \end{cases} \tag{V.54}$$

$$y_i^k \in \{0,1\} \quad (\forall\, i \in \mathcal{C}),\ k \in \mathcal{H} \tag{V.55}$$

$$\sum_{i \in \mathcal{C}} x_{ij}^k = y_j^k \quad (j \in \mathcal{C} + \{0\}),\ k \in \mathcal{H} \tag{V.56}$$

$$\sum_{j \in \mathcal{C}} x_{ij}^k = y_i^k \quad (i \in \mathcal{C} + \{0\}),\ k \in \mathcal{H} \tag{V.57}$$

$$x_{ij}^k \in \{0,1\} \quad \forall\, i,j\ \mathcal{C} + \{0\},\ k \in \mathcal{H} \tag{V.58}$$

$$t_j \geq t_i + s_i + t_{ij} - (1 - x_{ij}^k)T \quad \forall\, i,j \in \mathcal{C},\ k \in \mathcal{H} \tag{V.59}$$

$$a_i \leq t_i \leq b_i \quad \forall\, i \in \mathcal{C} \tag{V.60}$$

$$t_i \geq 0, \quad \forall\, i \in \mathcal{C} + \{0\} \tag{V.61}$$

O objetivo é minimizar os custos de percurso sujeitos à capacidade dos veículos e às restrições de tempo de chegada viável nos clientes. Pode-se distinguir três grupos de restrições na formulação de programação inteira acima. As restrições de V.53 - V.55 são aquelas de atribuição/agrupamento, relacionadas à atribuição viável de pedidos de clientes e dominadas pelas variáveis de atribuição y. Restrições de capacidade V.53 permitem que o frete total do cliente i possa ser levado pelo veículo k se ele está dentro de sua capacidade. As restrições V.54 consideram que todas as rotas iniciam e terminam no depósito e que cada cliente i é servido por apenas um veículo k.

O segundo grupo contém as restrições de roteamento V.56-V.58 as quais incluem as variáveis de decisão x que definem o percurso de cada veículo através de todos os clientes atribuidos a ele.

Finalmente as restrições V.59-V.61, incorporam as variáveis de tempo t que estão relacionadas às restrições de aprazamento (*scheduling*). O conjunto de restrições V.59, permitem compatibilizar tempos de chegada entre quaisquer dois clientes i e j, enquanto as restrições V.60 forçam o serviço nas janelas de tempo. Nota-se que as restrições V.59 necessitam ser definidas explicitamente para o depósito, desde que o tempo de chegada seja sempre maior do que o tempo de partida.

Dois problemas estão caracterizados na formulação do PPV-JT, o Problema de Atribuição/Agrupamento V.53-V.58, e o PCV com restrições de janelas de tempo V.59-V.61. Através desta observação, [Koskosides et al 92] propuseram um procedimento heurístico com o objetivo de resolver esta abordagem através de um procedimento semi-automático, utilizando o Problema de Agrupamento Capacitado (PAC), [Mulvey & Beck 84], [Koskosides & Powell 92], [Negreiros & Maculan 94], e soluções do PCV-JT.

O trabalho sobre o PPV-JT tem enfocado basicamente o uso de métodos heurísticos. Isto se deve à complexidade desta classe de problemas. Nos trabalhos de [Salvesbergh 84] e [Solomon 87] indica-se que o PPV-JT é bem mais difícil do que o PPV original. Não apenas o PPV-JT é $\mathcal{NP}$-Árduo, mas encontrar uma solução viável quando o tamanho da frota é fixado é por si só um problema $\mathcal{NP}$-Completo, [Salvesbergh 84].

Apesar de não aparecer nos modelos clássicos, admite-se que as restrições de janelas de tempo sejam violadas, penalizando na função objetivo aquelas soluções onde se chega mais cedo em determinados clientes (provocando um tempo de espera), [Salvesbergh 92].

Ao caso proposto onde as restrições com janelas de tempo não podem ser violadas, denomina-se de Problemas com Janelas de Tempo *Hard*. Enquanto que quando se admite um tempo de espera por violação das restrições tem-se os Problemas com Janelas de Tempo *Soft*.

Métodos exatos foram desenvolvidos para os problemas *hard*. Após o *survey* de [Solomon & Desrosiers 88], [Fisher et al 91] apresentaram resultados preliminares para duas aproximações exatas do PPV-JT, ambas usando relaxação lagrangeana. A primeira aproximação usa particionamentos variáveis para dividir o problema em um problema de semi-atribuição, e uma série de problemas de caminho mínimo com restrições de Janelas de Tempo. A segunda aproximação, é uma extensão do método de Fisher K-Árvore, com restrições de Janelas de Tempo sendo incorporadas na função objetivo no modo da relaxação lagrangeana. Am-

bas as aproximações consideram a presença de 25 a 50 clientes com janelas de tempo. [Fisher et al 91] são críticos com relação aos problemas testes propostos por [Solomon & Desrosiers 88], onde consideram que os problemas testes devam ser mais de acordo com a realidade.

[Desrochers et al 92] apresentaram uma aproximação que é capaz de resolver exatamente sete dos problemas com 100 clientes propostos por Solomon[2], dentre eles dois são dos grupos de instâncias R1 e cinco são de C1. Uma formulação de particionamento de conjuntos é usada, através de uma geração de colunas com o objetivo de gerar rotas melhores. A geração de colunas resolve um problema de caminho mínimo com janelas de tempo e restrições de capacidade, gerando várias colunas por camada de modo a reduzir o esforço computacional geral. Uma vez que a geração de colunas pode gerar rotas contendo ciclos, [Desrochers et al 92] resolveram um problema de cobertura de conjuntos ao invés de particionamento de conjuntos. Se necessário, a solução é forçada para uma solução do tipo particionamento de conjunto em um esquema tipo B&B. Devido ao algoritmo não poder resolver exatamente todos os problemas com 100 clientes, os autores propuseram a resolução inicial para dos primeiros 25 clientes, caso haja sucesso então passam para os próximos 50, e daí para o restante do conjunto.

A relaxação LP consegue excelentes limites inferiores para 27 dos 87 problemas. Nos outros grupos de instâncias, os limites estão a 1,5% da solução ótima. Através de uma variedade de esquemas de ramificação, a aproximação é capaz de resolver muitos problemas exatamente. Entretanto, ainda há problemas onde o algoritmo falha. Na média, cada problema que foi resolvido no ótimo necessitou 100 segundos de execução numa estação SPARC; entretanto este tempo variou consideravelmente.

V.4 O PPV com Repuxo de Carga

O Repuxo de Carga (*Backhaul*) é uma opção lucrativa e frequentemente usada nas aplicações reais de roteamento as quais envolvem a atividade de *pegar* uma carga no retorno de uma viagem ao depósito, após uma sequência de entregas realizadas. Esta atividade troca o trajeto sem serviço (*deadheading*) de volta ao depósito, com uma atividade de serviço lucrativa. Tipicamente, os repuxos ocorrem somente no final de uma rota, e são limitados a uma ou duas cargas.

Para tirar toda a vantagem das economias proporcionadas pelo repuxo de carga, as rotas regulares de entrega devem ser construidas de modo a incorporar os repuxos de carga com apenas um pequeno aumento no custo. Algumas das

[2]Maiores detalhes, consultar: `http://dmawww.epfl.ch/rochat/rochat_data/solomon.html`

técnicas de economias de inserção para o problema padrão de percurso de veículos foram modificadas por [Deif & Bodin 84], [Casco et al 88], e mais recentemente uma aplicação multi-depósito de [Min et al 92], de modo a atingir este objetivo.

Um modelo matemático para o problema foi proposto por [Goetschslckx & Jacobs-Blecha 89], considerando uma extensão da formulação proposta por [Fisher & Jaikumar 81] para uma frota homogênea, como segue.

Sejam dados,

K, número de veículos;

N, clientes sem repuxo, $N = 1, \ldots, n$;

M, clientes com repuxo, $M = n+1, \ldots, n+m$ (o índice 0 indica o depósito central);

a_i, demanda do cliente sem repuxo i;

b_i, quantidade suprida pelo cliente com repuxo i;

Q, capacidade fixa dos veículos de entrega;

c_{ij}, custo de viagem direta entre dois pontos i e j.

As variáveis são,

$$u_i^k = \begin{cases} 1, & \text{Se o cliente } i \text{ sem repuxo é servido pelo veículo } k; \\ 0, & \text{Caso contrário.} \end{cases}$$

$$v_j^k = \begin{cases} 1, & \text{Se o cliente } j \text{ com repuxo é servido pelo veículo } k; \\ 0, & \text{Caso contrário.} \end{cases}$$

$$x_{ij}^k = \begin{cases} 1, & \text{Se o veículo } k \text{ viaja do cliente } i \text{ ao cliente } j; \\ 0, & \text{Caso contrário.} \end{cases}$$

$$\text{(PPV-RC) Minimizar} \sum_k \sum_{i=0}^{n+m} \sum_{j=0}^{n+m} c_{ij} x_{ij}^k \tag{V.62}$$

Sujeito a:

$$\sum_{i=1}^{n} q_i u_i^k \leq Q \quad (k \in K) \tag{V.63}$$

$$\sum_{i=1}^{K} u_i^k = 1 \quad (i \in N) \tag{V.64}$$

$$u_0^k = 1 \quad (k \in K) \tag{V.65}$$

$$u_i^k \in \{0,1\} \quad (\forall\, i \in N),\ k \in K \tag{V.66}$$

$$\sum_{i=n+1}^{n+m} b_i v_i^k \leq Q, \quad (k \in K) \tag{V.67}$$

$$\sum_{k} v_i^k \leq 1, \quad (i \in M) \tag{V.68}$$

$$v_0^k = 1, \quad (k \in K) \tag{V.69}$$

$$v_i^k \in \{0,1\}, \quad (i \in M), (k \in K) \tag{V.70}$$

$$\sum_{i=0}^{n+m} x_{ij}^k = \begin{cases} u_j^k, & j \in N \\ v_j^k, & j \in M, j = 0, k \in K \end{cases} \tag{V.71}$$

$$\sum_{j=0}^{n+m} x_{ij}^k = \begin{cases} u_i^k, & i \in N \\ v_i^k, & i \in M, j = 0, k \in K \end{cases} \tag{V.72}$$

$$\sum_{i=0}^{n} \sum_{j \in M \text{ e } j=0} x_{ij}^k = 1, \ k \in K \tag{V.73}$$

$$x_{ij}^k \in S \tag{V.74}$$

$$x_{ij}^k \in \{0,1\} \quad (i,j = 0,\ldots,n+m)\ (k \in K) \tag{V.75}$$

As restrições V.63 e V.67 garantem que os veículos não serão sobrecarregados tanto sem repuxo quanto com repuxo de carga, respectivamente, em qualquer rota. As restrições V.64 e V.68 indicam que apenas um veículo pode ser atribuido a cada rota sem e com repuxo, respectivamente, e tal que não há mais do que K veículos saindo e retornando ao centro de distribuição. O conjunto de restrições V.71-V.73 prevê que apenas um veículo deverá atender a um cliente (saindo e entrando dele). As restrições V.74 são as restrições de eliminação de subrotas do PCV associado ao PPV-RC.

A limitação da composição dos serviços de entrega no PPV-RC torna-o um caso especial dos problemas gerais de coleta e entrega. Este problema é rotineiramente realizado nos serviços de entrega de malotes dos correios, que é bem mais difícil e ainda não foi explorado pela literatura, o Problema de Despacho/Coleta de Malotes, [Ochi & Negreiros 92].

O problema de Despacho/Coleta de Malotes, considera uma operação diária em duas fases:

1. Despacho de Malotes

- Demandas e destinos conhecidos;
- O problema é dinâmico, pois em dias subsequentes podem haver flutuações nas entregas e localização dos clientes;
- Os veículos partem da garagem e podem terminar seus percursos em agências próximas às regiões de trabalho (para almoço ou lanche);
- Horário de atendimento limitado ao turno matinal e parte da tarde.

2. Coleta de Malotes

- Demanda e destinos do dia conhecidos;
- Horário de atendimento restrito às últimas horas de serviço da tarde;
- Flutuação no tamanho da demanda e localização dos clientes;
- Disponibilidade de frota própria baseada na garagem central e em agências.

O objetivo é a geração do conjunto de rotas diárias de serviço, envolvendo o despacho (manhã) e a coleta (tarde) de malotes da ECT, para um conjunto mínimo de veículos próprios e/ou afretados, a fim de serem atendidas todas as ofertas e demandas, minimizando os custos operacionais, figuras V.6, V.7 e V.8.

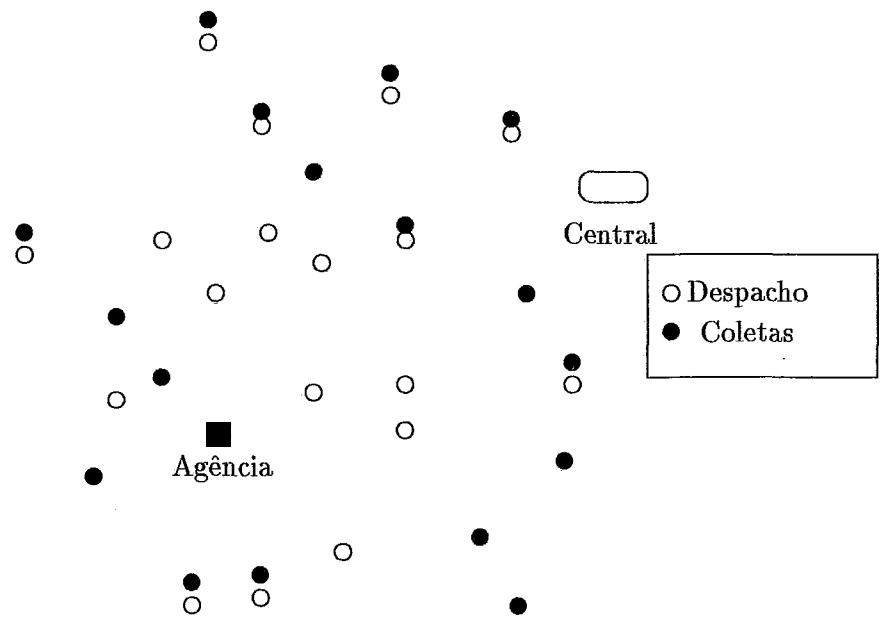


Figura V.6: Problema de Despacho/Coleta de Malotes.

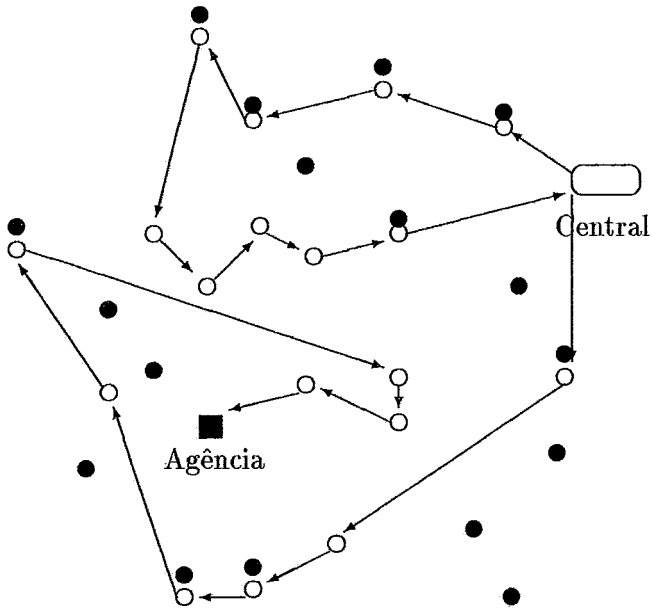


Figura V.7: Situação de Despacho de Malotes.

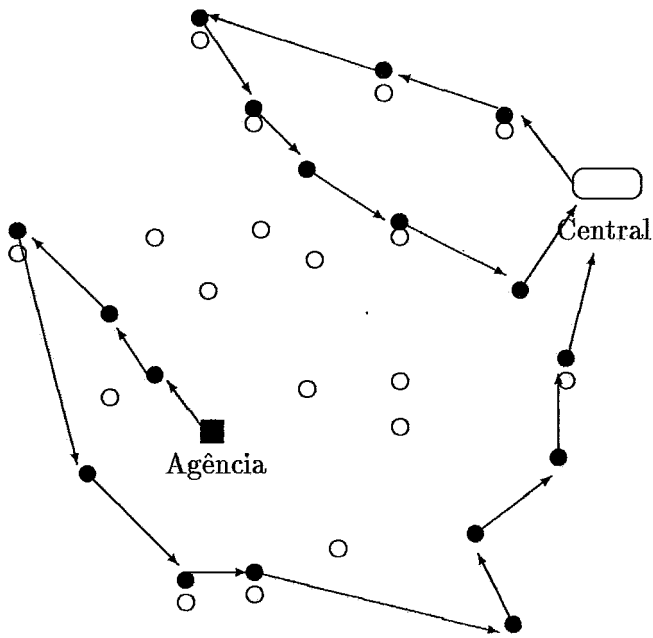


Figura V.8: Situação de Coleta de Malotes.

V.5 O PPV com Estoque e o PPV Periódico

Este problema acontece quando certos componentes de reposição de serviço são adicionados ao problema de roteamento. Neste problema, o tempo das entregas aos clientes devem ser determinados, baseados nas suas posições de estoque, na política de reposição, e nos custos de percurso. Muitos distribuidores de gases líquidos ou comprimidos possuem este problema uma vez que os níveis de estoque do cliente devem ser mantidos a uma quantidade adequada todo o tempo. Trabalhos importantes e variados sobre o assunto foram feitos por [Bell et al 83],[Federgruen & Zipkin 84], [Dror 83], [Golden et al 84a], [Trudeau & Dror 92], [Dror 93]. A estratégia apropriada para incorporar tempos longos de custos de estoque relacionados a tempos curtos dentro do mesmo modelo, e os problemas associados de validação na presença de previsões de demandas imperfeitas, são ainda problemas em aberto.

O PPV Periódico, considera a frequência de visita a cada cliente. Uma alternativa que se pode tomar no problema de estoques, é fixar a frequência para cada cliente sobre um determinado horizonte de planejamento, por exemplo uma semana. O objetivo é então atribuir dias da semana para as entregas do conjunto de clientes de modo a minimizar os custos semanais de distribuição enquanto atinge as necessidades de frequência. Por exemplo, um cliente que deseja três visitas por semana deve ser atribuido às Seg/Qua/Sex ou às Ter/Qui/Sáb. Este problema está muito próximo ao Problema do Motorista Vendedor, [Golden & Wasil 87], quando da distribuição de refrigerantes, cervejas e massas.

Algoritmos para o PPV Periódico geral foram propostos por [Russel & Igo 79], [Christofides & Beasley 84], [Tan & Beasley 84] e [Galdioso & Palleta 92]. No modelo geral proposto por [Galdioso & Palleta 92], supondo que um produto será entregue a partir de um depósito central a um conjunto de n clientes $I = \{1, \ldots, n\}$ geograficamente distribuidos. As entregas são feitas com o uso de veículos de igual capacidade Q, e um tempo máximo de serviço de H horas.

Cada cliente $i \in I$ necessita de uma entrega q_i $(0 \leq q_i \leq Q)$ do produto exatamente uma vez em cada intervalo de tempo de T_i dias num horizonte de tempo de planejamento de m dias.

Admite-se que é o menor múltiplo comum dos T_i's, portanto m/T_i é o número inteiro para todo $i \in I$ e representa o número de vezes que o cliente i é servido no horizonte de tempo de planejamento.

Impõe-se restrições adicionais onde duas entregas sucessivas devem ser espaçadas por pelo menos K_i, e no máximo U_i dias, $1 \leq K_i \leq T_i$, $T_i \leq U_i \leq 2T_i - 1$. Refere-se a qualquer conjunto de dias que satisfaz estas restrições como uma combinação de restrições viáveis para o cliente i.

Admite-se que um tempo de viagem (simétrico) t_{iu} é dado para cada par de clientes u; o depósito central é indexado por 0 e portanto t_0 é o tempo de viagem entre o depósito central e o i-ésimo cliente ($0 \leq t_0 \leq H/2$).

Durante qualquer dia j do período de planejamento, uma rota viável, (iniciando e terminando no depósito) é uma sequência ordenada de clientes (a ser visitado pelo mesmo veículo durante o dia j) satisfazendo às restrições:

- O tempo de viagem associado à sequência não deve exceder ao tempo H do veículo;
- A quantidade total de produto solicitado pelo cliente pertencendo à rota não deve exceder à capacidade de carga do veículo Q.

Nota-se que a princípio, é possível atribuir um mesmo veículo a mais de uma rota viável durante um mesmo dia. Portanto, define-se o serviço de um veículo durante qualquer dia j como o conjunto de rotas viáveis que podem ser atribuidas ao mesmo veículo sem violar o tempo de serviço diário H.

Devido à correspondência dos veículos em serviço ser de um-para-um com os veículos durante qualquer dia, usa-se de forma intercambiável os termos veículo em serviço ou simplesmente veículo.

Indica-se como tempo de serviço de um veículo durante qualquer dia, a soma dos tempos de viagem das rotas realmente atribuidas ao veículo durante dado dia.

O problema que se deseja resolver considera os seguintes pontos:

- Atribuir uma combinação de entrega viável a cada cliente;
- Atribuir, durante cada dia, uma rota viável para cada cliente a ser servido;
- Atribuir rotas aos veículos.

Enquanto minimiza o máximo número de veículos em operação durante o mesmo dia (minimizando o tamanho da frota).

Sejam dados,

$Y_j^r \equiv \{i : i \in I, x_{ij}^r = 1\}$ é o conjunto (possivelmente vazio) de clientes que pertencem à rota r do dia j.

$f(Y_j^r)$ é o tempo ótimo de viagem obtido pela solução do PCV sobre um conjunto de nós $Y_j^r \cup \{0\}$

R é um limite superior sobre um número de rotas definidas durante o mesmo dia (qualquer $R \geq n$ tem esta propriedade);

V é um limite superior sobre um número de veículos que operam durante o mesmo dia (qualquer $V \geq n$ tem esta propriedade).

Definindo-se as varáveis do problema, tem-se:

$$x_i^j = \begin{cases} 1, & \text{Se o cliente } i \text{ é servido no dia } j; \\ 0, & \text{Caso contrário.} \end{cases}$$

$$x_{ij}^r = \begin{cases} 1, & \text{Se o cliente } i \text{ pertence ao dia } j \text{ na rota } r; \\ 0, & \text{Caso contrário.} \end{cases}$$

$$y_j^{rv} = \begin{cases} 1, & \text{Se a rota } r \text{ do dia } j \text{ está atribuida ao veículo } v; \\ 0, & \text{Caso contrário.} \end{cases}$$

$$y_j^r = \begin{cases} 1, & \text{Se a rota } r \text{ é usada no dia } j; \\ 0, & \text{Caso contrário.} \end{cases}$$

$$z_j^v = \begin{cases} 1, & \text{Se o veículo } v \text{ é usado durante o dia } j; \\ 0, & \text{Caso contrário.} \end{cases}$$

$$\text{(PPV-Periódico) Minimizar } z \tag{V.76}$$

Sujeito a:

$$\sum_{j=(k-1)T_i+1}^{kT_i} x_i^j = 1 \quad (i = 1, \ldots, n) \; (k = 1, \ldots, m/T_i) \tag{V.77}$$

$$\sum_{j=h}^{h+K_i-1} x_i^j \leq 1 \quad (i = 1, \ldots, n) \; (h = 1, \ldots, m - K_i + 1) \tag{V.78}$$

$$\sum_{j=h}^{h+U_i-1} x_i^j \geq 1 \quad (i = 1, \ldots, n) \; (h = 1, \ldots, m - U_i + 1) \tag{V.79}$$

$$\sum_{r=1}^{R} x_{ij}^r = x_i^j \quad (i = 1, \ldots, n) \; (j = 1, \ldots, m) \tag{V.80}$$

$$\sum_{i=1}^{n} x_{ij}^r \, q_i = Q \, y_j^r \quad (j = 1, \ldots, m) \; (r = 1, \ldots, R) \tag{V.81}$$

$$\sum_{v=1}^{V} y_j^{rv} = y_j^r \quad (j = 1, \ldots, m) \; (r = 1, \ldots, R) \tag{V.82}$$

$$\sum_{v=1}^{V} y_j^{rv} f(Y_j^r) \leq H z_j^v \quad (j = 1, \ldots, m)\ (v = 1, \ldots, V) \tag{V.83}$$

$$\sum_{v=1}^{V} z_j^v \leq z \quad (j = 1, \ldots, m) \tag{V.84}$$

As restrições V.77 indicam que o cliente i deve ser servido somente uma vez dentro de cada intervalo de tempo. As restrições V.78 e V.79 garantem a satisfação das necessidades de espaçamento entre serviços. As restrições V.80 permitem que cada cliente servido no dia j seja atribuido a exatamente uma rota. As restrições V.81 garantem que os parâmetros de carga de veículo são respeitados pelas rotas. As restrições V.82 admitem a atribuição de cada rota a exatamente um veículo. As retrições V.83 garantem que o conjunto de rotas atribuido a cada veículo são compatíveis com o tempo diário de serviço. As restrições V.84 são necessárias para transformar o problema min-max em um problema com função objetivo linear.

Nota-se do modelo, de acordo com o que se admitiu, que o problema é viável. Entretanto, as variáveis que satisfazem x_i^j nas restrições V.77,V.78 e V.79 são fixas, o problema pode ser decomposto em um conjunto de m problemas de roteamento de veículos cada um deles sendo definido sobre um conjunto de clientes $I_j = \{i : i \in I; x_i^j = 1\}$, $j = 1, \ldots, m$. De fato para cada valor de $j = 1, \ldots, m$ o problema é de minimizar $\sum_{v=1}^{V} z_j^v$ sujeito às restrições de V.78-V.83, ou seja, minimizar o número de veículos necessários para satisfazer o conjunto de usuários I_j, sobre as restrições impostas pela capacidade de carga e do tempo de serviço.

Além do mais, sempre que as variáveis x_i^j, x_{ij}^r e y_j^r estão fixadas (e consequentemente $f(Y_j^r)$ são admitidas ser conhecidas) o problema é decomposto a um conjunto de problemas *m-binpacking*, onde H é a dimensão da *bin* e $f(Y_j^r)$ é o tamanho do item r. (Aqui, para cada j, $j = 1, \ldots, m$ a função objetivo a ser minimizada $\sum_{r=1}^{v} z_j^v$ está novamente sobre as restrições V.82 e V.83).

Trocando o valor de $f(Y_j^r)$ definido acima com $f(Y_j^r)$, o tempo de viagem associado ao caminho hamiltoniano definido sobre $Y_j^r \cup \{0\}$, chega-se à formulação que permite adotar técnicas heurísticas para resolver o PCV embutido no problema. O problema é $\mathcal{NP}$-Árduo pela reducibilidade ao problema de *binpacking*.

Considerando a modelagem proposta, a literatura não tem explorado algoritmos exatos na sua solução. [Galdioso & Palleta 92], propuseram uma heurística com uma fase construtiva baseada nas seguintes características:

1. Um algoritmo de enumeração implícita das combinações viáveis de entrega aos clientes i_j;

2. Um algoritmo heurístico para atribuição de clientes a rotas e para a deter-

minação das rotas;

3. Um algoritmo de empacotamento para a atribuição de veículos a rotas.

Várias instâncias foram geradas com 100 a 300 clientes, e horizontes de tempo de 6 a 12 *dias*. Em geral os tempos computacionais mostraram-se muito razoáveis, porém nenhuma comparação a outros métodos da literatura foi apresentada.

V.6 O PPV com Composição de Frota

Os problemas anteriormente citados são em geral operacionais. Uma empresa que está envolvida com distribuição precisa se preparar também com relação a situações emergenciais. Um exemplo importante, é a decisão sobre o tamanho da frota e sua composição. Os veículos na frota devem ser de diferentes tipos e capacidades, no problema de frota e composição, deseja-se determinar a composição da frota que minimiza a soma dos custos fixos, associados aos custos de aquisição e aluguel (*leasing*), e os custos de depreciação. Os custos de percurso podem ser calculados pelo uso de uma frota disponível com o objetivo de resolver o problema de roteamento (diário) para um grande número de vezes. Este problema foi estudado por [Etezadi & Beasley 83], [Gheysens et al 86], [Golden et al 84b], [Salhi et al 92].

A formulação matemática do problema proposta por [Salhi et al 92], é colocada como segue.

Sejam dados,

N, o conjunto de clientes $(1, \ldots, n)$ e n_0 o depósito;

K, é o conjunto de tipos de veículos $(1, \ldots, k)$;

F_k, é o custo fixo do veículo tipo k;

Q_k, é a capacidade do veículo tipo k;

T_k, é o tempo máximo para o veículo tipo k;

q_i é a demanda do i-ésimo cliente.

t_{ij}^k é o tempo de viagem do cliente i ao j com o veículo k;

c_{ij}^k é o custo de viagem do cliente i ao j com o veículo k;

e como variáveis tem-se,

$$x_{ij}^k = \begin{cases} 1, & \text{Se o veículo } k \text{ é usado entre os clientes } i \text{ e } j \\ 0, & \text{Caso contrário.} \end{cases}$$

y_{ij}, é uma variável contínua que determina a quantidade de produto conduzida entre os clientes i e j.

R_{ij}, é uma variável contínua que denota o tempo que ainda resta após a cobertura da ligação ij.

$$\text{(PPV-Mix Frota) Minimizar } \sum_{k\in K} F_k \sum_{j\in N} x_{0j}^k + \sum_{k\in K}\sum_{i,j\in N} c_{ij}^k x_{ij}^k \tag{V.85}$$

Sujeito a:

$$\sum_{k\in K}\sum_{i=0}^{n} x_{ij}^k = 1 \quad (j = 1,\ldots,n) \tag{V.86}$$

$$\sum_{i=0}^{n} x_{ij}^k - \sum_{i=0}^{n} x_{ji}^k = 0 \quad (j \in N + \{0\})\ (k \in K) \tag{V.87}$$

$$\sum_{i=0}^{n} y_{ij} - \sum_{i=0}^{n} y_{ji} = q_i \quad (j \in N + \{0\}) \tag{V.88}$$

$$\sum_{j=1}^{n} y_{0j}^k \leq \sum_{i=1}^{n} q_i \tag{V.89}$$

$$\sum_{i=1}^{n} y_{i0} = 0 \tag{V.90}$$

$$y_{ij} \leq \sum_{k=1}^{K} x_{ij}^k Q_k \quad \forall\, j \neq i \tag{V.91}$$

$$R_{ij} \leq \sum_{k=1}^{K} T_k x_{ij}^k \quad \forall\, i \neq j \tag{V.92}$$

$$R_{0j} = \sum_{k=1}^{K} x_{0j}^k T_k - \sum_{k=1}^{K} x_{0j}^k t_{0j}^k \quad (j \in N) \tag{V.93}$$

$$\sum_{i=0}^{n} R_{ip} - \sum_{i=0}^{n} R_{pj} = \sum_{k=1}^{K}\sum_{j=0}^{n} x_{pj}^k t_{pj}^k \quad (p \in N) \tag{V.94}$$

$$\left.\begin{array}{l} x_{ij}^k \in \{0,1\} \\ y_{ij} \geq 0,\ y_{ii} = 0 \\ R_{ij} \geq 0 \end{array}\right\} (i,j \in N \cup \{0\})\ (k \in K) \tag{V.95}$$

Em V.85 a função objetivo considera a minimização dos custos de

utilização da frota e de percurso. As restrições V.86 mostram que cada cliente é visitado somente uma vez e V.87 permite a consevação de fluxo em cada cliente. As restrições de V.88 a V.91 referem-se às restrições de capacidade e eliminação de subrotas considerando as idéias de fluxo com 1-comodidade. As restrições de V.92-V.94 consideram o tempo que resta.

[Salhi et al 92] propõem algoritmos heurísticos de melhoria baseados em particionamento e pertubação de rotas os quais ultrapassam os resultados alcançados por [Golden et al 84b].

Uma outra abordagem, deste problema, segue-se na tomada de decisão relacionada à determinação do número de veículos a alugar (*leasing*), dado que exista uma opção de reutilização de veículo no meio de rotas longas, [Bodin et al 83].

V.7 O PPV com Demanda Dividida

Na essência, como já se viu, o PPV não permite que um veículo seja servido por mais de um veículo. Alguns pesquisadores, porém, estudaram uma relaxação do PPV, no que concerne a possibilidade de divisão da demanda de um cliente pelo atendimento de veículos distintos. A este problema denominou-se o PPV com Demanda Dividida - *Split Delivery*, PPV-DD.

[Burrows 88] modificou o algoritmo de economias de modo a permitir a divisão da demanda. A modificação feita releva o fato de que o pedido de um cliente é dividido em um número de pedidos indivisíveis. O algoritmo de economias é então aplicado sobre o conjunto total de vértices particionados. [Dror & Trudeau 90] adaptaram um algoritmo para o PPV desenvolvido por [Dror & Levi 86], onde dois novos procedimentos de melhoria surgiram:

1. Mudança em k-Divisões: Divide uma ordem de serviço em k porções, cada uma servida por um veículo diferente;

2. Adição de Rota: Uma rota extra é adicionada onde cada uma delas elimina a divisão para um cliente particular.

Os resultados apresentados por [Dror & Trudeau 90] ilustram as vantagens de permitir a divisão de entregas/coletas. Quando pedidos que chegam de 70-90% da capacidade dos veículos, uma melhoria acima de 10% na distância total foi obtida sobre várias instâncias do PPV.

[Pedder 90] desenvolveu um sistema de roteamento e planejamento de frotas para uma cadeia de supermercados, onde no problema básico os pedidos

dos clientes são sempre próximos, ou excedem a capacidade dos veículos. Foi então desenvolvido um modelo que permitiu dividir as entregas, baseando-se nas idéias de [Foster & Ryan 76] exceto que uma simples coluna poderia representar várias rotas.

[Dror et al 94] tentaram resolver o PPV-DD exatamente, usando relaxação de restrições com B&B. Para as instâncias consideradas, os limites inferiores obtidos ficaram dentro dos 9% dos limites superiores. Somente uma das instâncias foi resolvida no ótimo. Em geral, as instâncias tomadas eram pequenas, 10-20 clientes. Uma conclusão importante foi reportada pelos autores considerando que a dificuldade do PPV-DD é superior à do PPV original.

V.8 O PPV com Múltiplos Compartimentos

Até aqui os PPV especiais estão relacionados a situações onde existem veículos que atendem seus clientes com um único produto e um único local de armazenamento de modo a cumprir o atendimento da série de clientes com demandas previamente conhecidas.

O surgimento das aplicações com múltiplos-compartimentos de carga, veio da necessidade de se construir rotas que atendessem a um vasto conjunto de restrições as quais se encaixam sobre as necessidades operacionais de serviço de vários clientes preferenciais que ocorrem na distribuição de derivados de petróleo e combustíveis, e no roteamento de navios tanques ou graneleiros, [Golden et al 84a], [Brown & Graves 81], [Bell et al 83], [Brown et al 87], [Bausch et al 91], [Negreiros & Negreiros 94].

Uma grande variedade de situações de despacho acontecem neste caso específico, através do mundo, [Ronen 95], [Negreiros & Negreiros 94]. Dentre elas tem-se:

- Um ou muitos produtos são enviados a um mesmo cliente. Os terminais de carga (tanques) devem despachar separadamente, ou uma área que contém múltiplas fontes devem ser despachadas juntas. No último caso, um cliente pode ser suprido por diferentes terminais, podendo haver diferenças de custo entre eles para um mesmo produto. A troca de produto entre as companhias de óleo permitem que uma companhia atraia produtos de muitos terminais.
- Os suprimentos de certos produtos está limitado a certas fontes.
- Os caminhões tanques podem realizar múltiplas rotas por período.
- Um caminhão pode ter até nove paradas (ou mais) numa única rota;

- Um caminhão pode carregar em mais de um terminal produtos diferentes para uma mesma viagem.
- Pode haver muitas entregas no mesmo cliente num mesmo dia (tanto com produtos diferentes, quanto com grandes quantidades).
- Caminhões não têm medidores para aferir as quantidades entregues (todo o compartimento é esvaziado com o objetivo de apressar a entrega). A quantidade entregue a um cliente é medida no momento de carga do caminhão.
- Caminhões devem ser carregados pelo volume de produto ou o seu peso (esta provê uma melhor utilização da capacidade do caminhão). Quando o produto é quente (tem baixa densidade), um volume alto deve ser carregado sobre um veículo sem violar os limites de peso (devido à lei da balança nas estradas).
- Uma separação especial entre certos produtos sobre um veículo deve ser feita de modo a prevenir a contaminação devido à troca de produtos no caso de haver perfuração nas paredes. Se tal separação não existir, um compartimento livre deve existir entre os líquidos.
- Ajuste de ordens de quantidade de produto (de modo a se adequar aos compartimentos do caminhão), deve ou não ser permitida (sobre certos limites).
- Janelas de tempo específicas devem, ou não, ser utilizadas. As janelas de tempo podem ser *hard* ou *soft*, largas ou estreitas, e uma entrega deve ter uma ou mais janelas de tempo.
- Primeira/última carga (no dia) devem ser atribuídas considerando uma ordem na compartimentação. Além disso é importante considerar que muitos clientes desejam suas entregas no início ou no final do dia, neste caso é difícil o atendimento de todos.
- Balanceamento nas cargas dos veículos (horas e número de viagens) são necessárias de modo a atender as restrições de pagamento haja vista os requisitos contratuais.
- Alguns clientes podem ter seus tanques colocados em posições adversas, onde o único modo de preenchê-los é com o uso de bombas. Neste caso, apenas alguns veículos são equipados com este tipo de equipamento.
- A frota é em geral heterogênea (compartimentos distintos em número e capacidade entre veículos).
- Três tipos de operação podem surgir, as operações em *city market* consideram que os clientes estão dentro de uma região de atendimento onde não incidem custos variáveis sobre o preço do frete. A operação fora de *city market* nas

situações onde os custos incidem sobre as distâncias percorridas, e são calculados em escala relativamente à distância da base de carga. Por fim, a operação mista considera a possibilidade de ultrapassar a região de *city market*, porém permitindo que esta ultrapassagem não exceda certos limites dentro do dia de trabalho.

- Nas situações de distribuição em países continentais como o Brasil, tem-se em torno de 40 terminais para 5000 postos, em uma das empresas distribuidoras que despontam entre as maiores do país.

- A tancagem típica dos veículos compartimentados é de 10 a $35m^3$, enquanto que as demandas diárias variam de 1 a $10m^3$ por cliente por produto. Os tanques são geralmente divididos em compartimentos de 2, 3, 4, 5, 6, 7 e $10m^3$.

Em face da variedade de restrições existentes no problema, foi proposto um método de geração de colunas, chamado Particionamento de Conjunto Elástico (PCE), [Brown et al 87]. Este método é baseado em um modelo de programação inteira que é uma extensão do problema de Particionamento de Conjuntos. Esta classe de problemas têm sido usadas com frequência no planejamento de transportes por diversos autores, como já foi visto em seções anteriores. Em geral, porém, os autores têm que recorrer a heurísticas sem permitir soluções ótimas ou sem saber qual a qualidade da solução obtida. A vantagem do modelo de particionamento é que ele pode acomodar custos discretos e não lineares, frotas heterogêneas, e uma grande variedade de regras de operação de frota. Isto pode ser atingido pela geração de um grande número (ou todos) os planos para cada unidade de transporte e selecionar um plano geral para cada unidade de transporte de modo que minimize os custos envolvidos no trasporte para realização de todas as tarefas solicitadas. A grande desvantagem é que estes modelos são difíceis de serem resolvidos no ótimo.

No problema elástico, os autores permitem que sejam violadas as restrições do conjunto de particionamento sobre uma certa penalidade, e a função objetivo minimiza os custos dos planos mais o valor das penalidades violadas. O PCE é uma formulação mais compacta do problema, uma vez que certas restrições não precisam ser explicitamente colocadas.

A formulação do problema é a que segue.

Sejam dados,

$o \in O$, são as ordens de serviço (entregas/coletas);

$t \in T$, tipos de veículos;

$p \in P$, planos alternativos de serviço;

P_0, planos que incluem a ordem 0;

P_t, planos que necessitam do veículo tipo t;

C_p, custo do plano p;

$\overline{OP}_o, \underline{OP}_o$, penalidades de *sobre* ou *sub* atendimento à ordem o;

NT_t, número de veículos do tipo t disponíveis;

$\overline{TP}_t, \underline{TP}_t$, penalidades de *sobre* ou *sub* utilização do veículo t;

$$y_p = \begin{cases} 1, & \text{Se o plano } p \text{ é usado} \\ 0, & \text{Caso contrário.} \end{cases}$$

$\overline{\omega}_o, \underline{\omega}_o$, variáveis para controle das restrições elásticas no *sobre* ou *sub* atendimento da ordem o;

$\overline{\sigma}_t, \underline{\sigma}_t$, variáveis para controle das restrições elásticas na *sobre* ou *sub* utilização do veículo tipo t.

$$\text{(PPCE) Min} \sum_p c_p y_p + \sum_o (\underline{OP}_o \underline{\omega}_o + \overline{OP}_o \overline{\omega}_o) + \sum_t (\underline{TP}_t \underline{\sigma}_t + \overline{TP}_t \overline{\sigma}_t) \quad \text{(V.96)}$$

Sujeito a:

$$\sum_{p \in S_o} y_p + \underline{\omega}_o - \overline{\omega}_o = 1 \quad (o \in O) \quad \text{(V.97)}$$

$$\sum_{p \in S_t} y_p + \underline{\sigma}_t - \overline{\sigma}_t = NT_t \quad (\forall\, t \in T) \quad \text{(V.98)}$$

$$y_p \in \{0, 1\} \quad (\forall\, p \in P) \quad \text{(V.99)}$$

$$(\underline{\omega}_o, \overline{\omega}_o) \geq 0, \quad \forall\, o \in O \quad \text{(V.100)}$$

$$(\underline{\sigma}_o, \overline{\sigma}_o) \geq 0, \quad \forall\, t \in T \quad \text{(V.101)}$$

As restrições V.97 procuram escalonar uma ordem de serviço, uma violação inferior representa a necessidade de um redespacho intermediário, enquanto que uma violação superior representa uma grande ruptura na penalidade (que deve ser analisada pelo despachante quando ocorrer). Cada restrição V.98 procura por um plano por um veículo tipo p, onde a violação inferior representa a falta de necessidade destes veículos, e uma violação superior representa uma ruptura na penalidade, sugerindo que mais veículos deste tipo são necessários. As restrições V.99, V.100 e V.101 especificam que a seleção do plano e as variáveis elásticas lógicas são não negativas. A função objetivo V.96 inclui então os custos de transporte de seleção dos planos assim como a influência das penalidades.

Cada plano alternativo S está associado a um veículo tipo t, e se converte a um conjunto de ordens de paradas sequenciadas. A derivação de c_p inclui encontrar para cada carregamento de veículo viável, a combinação de ordens a uma distância mínima viável de sequência a qual pode ser conduzida a uma rota, com entregas ponto-a-ponto e coletas. Se o veículo tipo t retorna a sua localização inicial, múltiplas rotas podem ser concatenadas em um dia. Cada rota é calculada, trecho a trecho, estando em c_p o valor do número total de todas as viagens e paradas no plano do dia.

No sistema utilizado pelos autores para a resolução de grandes instâncias do problema, foram empregadas metodologias de solução não tradicionais tais como: restrições elásticas (como foi visto), fatorização de linhas, solução de problema em cascata, pré-redução, enumeração de restrições por ramificação, e um procedimento eficiente dual-simplex.

As restrições elásticas podem ser violadas a um dado custo linear de penalidade por unidade de violação. Cada restrição proposta é elástica. Já a fatorização de linhas, identifica e explora conjuntos de restrições as quais incorporam uma estrutura especial comum. As restrições V.98 têm pelo menos um coeficiente-unidade associado a cada variável, e assim qualifica os "limites superiores gerais" do problema. O uso desta fatorização reduz o tempo computacional de forma drástica.

Sempre as relaxações de programação linear do PPCE podem dificultar na solução direta. Soluções de problemas em cascata permitem que num modelo difícil a ser resolvido de forma incremental considerem: uma sequência de sub-modelos seja resolvida, sub-soluções sejam analisadas, e registros sejam mantidos para todas as execuções para cada restrição e variável, e as variáveis as quais seriam contrárias não fariam parte de um sub-modelo mantida no último valor conhecido. Eventualmente, o registro de variáveis podem ser usadas como um ponto de início avançado para resolver o modelo inteiro.

Na pré-redução das instâncias do modelo prioritariamente a otimização, ou seja, procurar e remover aspectos estruturalmente redundantes, que podem revelar a estrutura escondida do problema e tornar mais rápido o cálculo.

Ramificação das restrições é uma variação da enumeração B&B a qual seleciona uma variável sobre a base de sua influência direta e os efeitos indiretos dos valores que eles iriam induzir por outras variáveis estruturalmente dependentes. As ramificações de restrições aumentam a velocidade da enumeração inteira. As variáveis de ramificação são selecionadas pelas restrições baseadas em uma estimativa das consequências totais sobre os custos das restrições (ou seja, há um benefício na interação entre elasticidade e ramificação das restrições).

Por fim, o modelo tem bem menos restrições que variáveis, e o procedimento dual-simplex o resolve 10 vezes mais rápido do que o método primal.

Os autores reportam o uso de um sistema extremamente integrado entre a aquisição de dados diário, a análise de despacho através das ordens recebidas e restrições operacionais momentâneas que vem sendo utilizado por algumas empresas de distribuição de combustíveis através do mundo.

Uma vez que o problema multicompartimentado é bastante restrito, um número enumerável de planos é possível de ser realizado. Porém, em situações onde as restrições operacionais e de planejamento são reduzidas, o problema torna-se mais complexo de ser resolvido exatamente.

Uma das idéias ainda não exploradas pela literatura diz respeito ao uso de um mesmo compartimento para mais de um cliente, m-compartimento. Na opção de distribuição de derivados de petróleo e combustíveis esta tarefa é complicada, uma vez que no processo de entrega do produto será necessário que se equipe os veículos com medidores de vazão calibrados, aumentando o custo do sistema. A redução no custo de transporte, porém, compensaria rapidamente o investimento realizado, [Negreiros & Negreiros 94]. Esta metodologia também facilita na distribuição de produtos onde a *paletização* (arrumação dos veículos em compartimentos onde somente é armazenado um tipo de produto) dos veículos é um procedimento comum, e onde se admite a quebra de demanda podendo um mesmo cliente ser atendido por mais de um veículo.

Neste contexto, [Negreiros & Negreiros 94] apresentaram um método baseado em economias, que inicialmente desagrega demandas de clientes de acordo com o tamanho de compartimento a que se destinam as ordens de serviço a serem conduzidas. Nesta fase, o procedimento procura encontrar a quebra das ordens de acordo com o tamanho do menor compartimento, supondo que as ordens podem ter um tamanho maior que toda a soma de capacidades dos compartimentos.

Com as demandas quebradas, atribui-se a cada uma delas um rótulo de demanda de um dado cliente, em seguida aplica-se o método de economias com restrições na formação da composição da carga nos compartimentos, ou seja, produtos distintos devem estar em compartimentos distintos; buscando-se explorar o preenchimento dos compartimentos abertos e a seguir os não abertos. Esta metodologia reduz drasticamente o número de viagens em relação ao caso 1-compartimento/1-cliente/1-produto, ou 1-compartimento, à medida que os compartimentos são maiores em relação às demandas, [Negreiros & Negreiros 94].

O método de [Negreiros & Negreiros 94] propõe também a avaliação dos custos de transporte sobre o produto, uma vez que entre estados existe uma diferença nas taxas de impostos incidentes sobre a circulação de mercadorias e que dependem fortemente do local onde o cliente está relativamente aos terminais supridores mais próximos. Estas diferenças de taxas provocam que em situações de limites entre estados um cliente possa ser servido por outro estado, mesmo que haja uma

base localizada no seu estado. Isto acontece devido ao custo de transporte global considerar todos os fatores que influenciam o custo de transportar os produtos da fonte ao destino e retornar. Nestes fatores incluem as taxas incidentes sobre a circulação do produto. [Ronen 95] colocou este quadro como um preço diferenciado entre bases para um mesmo produto, de acordo com o cliente.

Um exemplo de aplicação de uma situação menos restritiva, é considerado como segue.

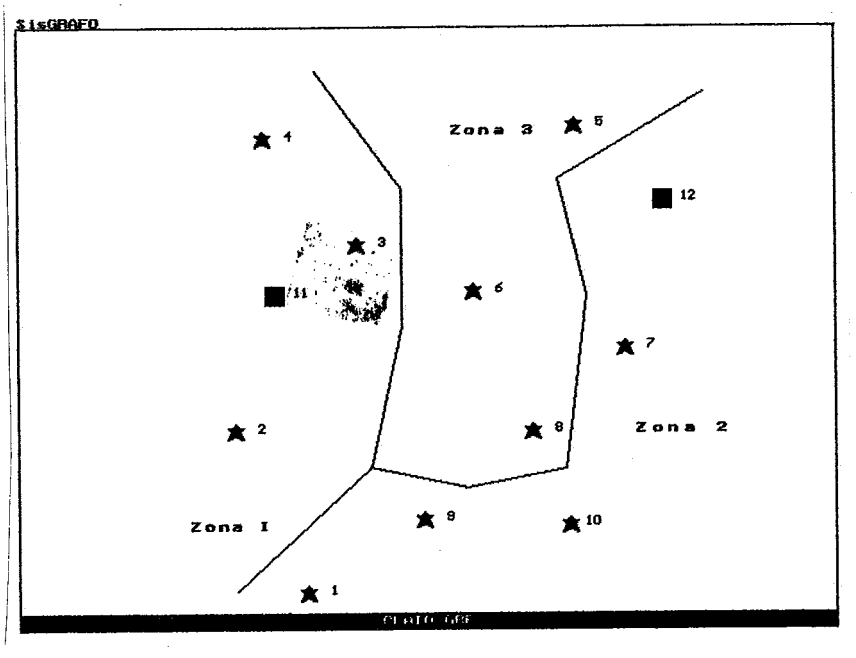


Figura V.9: Situação geográfica para um PPV-MC.

Dados que clientes estão espaçados em regiões geográficas definidas e supondo que cada unidade de distância percorrida corresponde a 0,55 unidades monetárias de custo, figura V.9. Nas tabelas V.2, V.3 e V.4 tem-se respecticamente as ordens de serviço por cliente, os custos por unidade de produto e os percentuais de impostos de circulação de mercadoria que incidem sobre o preço dos produtos. Sendo de K unidades a capacidade do compartimento típico de uma frota homogênea com c compartimentos por veículo, onde os vértices 11 e 12 são os terminais de carga, deseja-se conhecer as rotas de composições 1-compartimento e m-compartimento.

A tabela V.5 mostra o quanto se pode melhorar uma solução a partir da solução 1-compartimento mantendo-se constante o número de compartimentos e variando a tancagem dos veículos, permitindo que mais de um cliente possa ser atendido por um mesmo compartimento. Nas figuras V.10 e V.11 tem-se uma com-

Tabela V.2: Coordenadas e demandas por produto por cliente.

	Coordenadas		Produtos	
Clientes	X	Y	1	2
1	197	419	3	-
2	127	303	4	-
3	246	170	2	-
4	154	93	1	5
5	459	84	3	4
6	360	203	2	2
7	507	243	-	2
8	417	303	3	4
9	311	367	4	2
10	457	371	-	3
11	165	206	-	-
12	545	137	-	-

Tabela V.3: Custo por unidade de produto.

Produto	Custo
1	12
2	7

Tabela V.4: Impostos (%) por unidade de produto.

Zona	1	2	3
1	17	12	7
2	12	17	7
3	7	7	17

Tabela V.5: Comparação entre as soluções obtidas do exemplo.

Solução	K	c	No. Rotas	Distância	$ Total
1-Compartimento	5	3	5	1883,40	2578,07
m-Compartimento	5	3	4	1775,81	2381,44
m-Compartimento	8	3	3	1477,19	1839,50
m-Compartimento	10	3	3	1477,19	1839,50

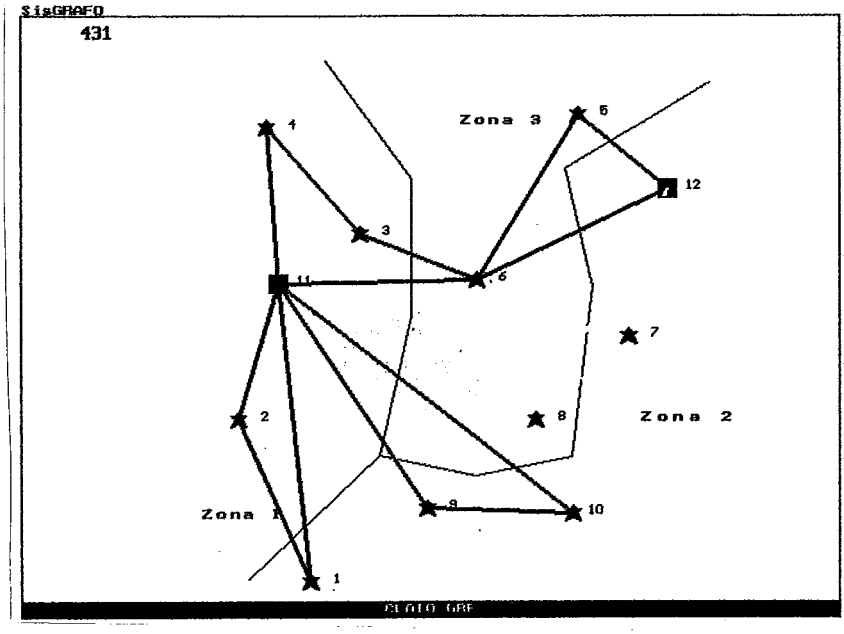


Figura V.10: Solução 1-Compartimento, $K = 5$ e $c = 3$.

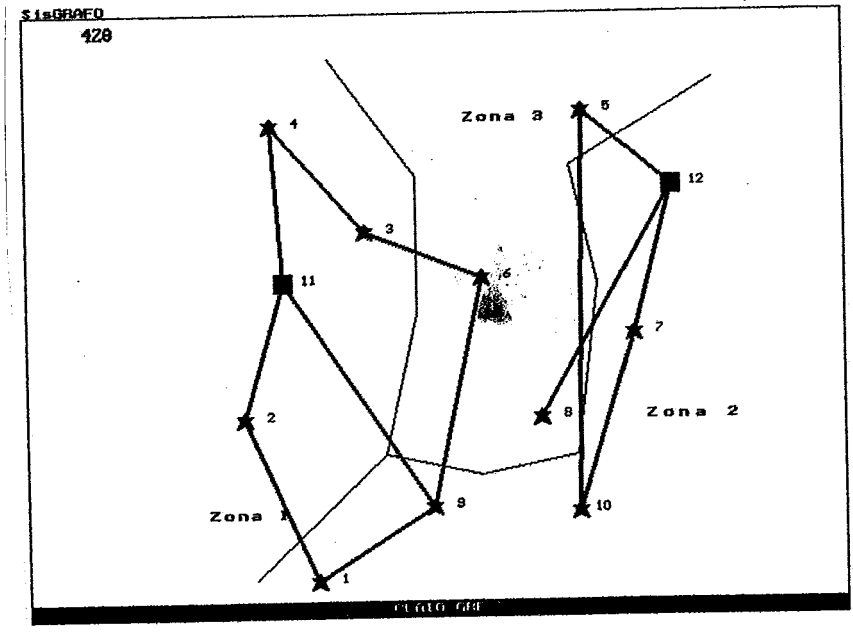


Figura V.11: Solução m-Compartimento, $K = 5$ e $c = 3$.

paração entre as situações onde apenas se permitiu que mais de um cliente fosse atendido por um mesmo compartimento, mantendo-se o mesmo conjunto de clientes.

Testes deste modelo foram analisados para uma instância com dados reais de planejamento fornecidos por uma empresa de distribuição de derivados de petróleo para cidades localizadas nos estados de São Paulo, Paraná e Santa Catarina. Neste problema, há três bases de distribuição localizadas em Curitiba (PR), Paranagua (PR) e Itajaí (SC), e 85 cidades cada uma solicitando três produtos (álcool hidratado, gasolina e diesel), os quais são enviados em pedidos mínimos de $1m^3$, para um plano de entrega equivalente a 348 atendimentos, figura V.12.

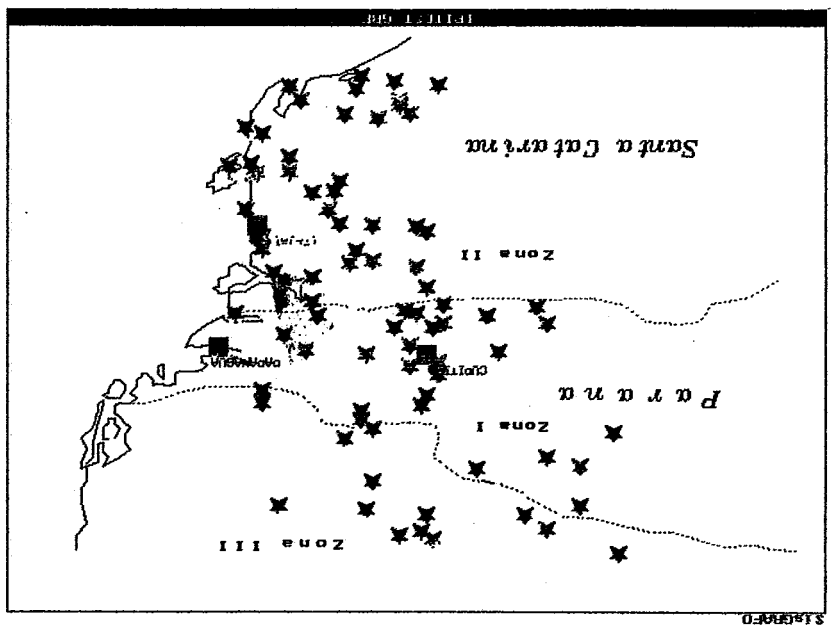


Figura V.12: Situação geográfica para um PPV-MC real.

Apesar dos dados de demanda e posicionamento dos clientes serem reais, uma análise teórica sobre uma situação de movimentação dos veículos, considerando os casos 1-compartimento e m-compartimento, foi feita com o objetivo de se ter de forma mais clara informações sobre o impacto da mudança de paradigma sobre os custos no sistema de distribuição. Na tabela V.6 nota-se também que a redução na distância (admitida euclidiana) foi de aproximadamente 6% e de 1% sobre os custos totais de transporte considerando o valor dos produtos. O tempo de processamento (TP) do algoritmo proposto para o número de atendimentos (quebras) mostrou-se com maior impacto na etapa de formação das economias (*savings*).

Como em geral os custos de manutenção de um veículo são geralmente

Tabela V.6: Comparação entre as soluções obtidas em instâncias com dados reais de demanda, custos de produto e transporte, e posicionamento de clientes.

Solução	Quebras	K	c	Viagens	Distância	$ Total	TP	*Savings*
1-Comp	348	8	4	131	11.002,64	377.983,03	482,57	67525
m-Comp	348	8	4	68	10.331,34	374.085,74	485,70	67525

altos, e considerando também uma frota típica alugada os custos de utilização são também altos. A mundança de paradigma é, pois, uma vantagem a ser fortemente observada.

Capítulo VI

O Problema de Percurso em Arcos

Os problemas de percursos em arcos são dos mais antigos relacionados a grafos. A primeira referência que se conhece sobre eles vem do famoso problema das sete pontes de Königsberg, figura VI.1. Os habitantes desta cidade tinham um desafio: perguntavam-se se existe um caminho fechado que atravessa exatamente uma vez as sete pontes sobre o rio Pregel em Köningsberg, hoje Kaliningrad. O problema foi resolvido pelo matemático suíço Leonhard Euler, que encontrou as condições para a existência de uma rota fechada, e mostrou que não havia nenhuma solução que satisfizesse aquele caso particular, [Euler 1736]. A preocupação de Euler foi exclusivamente sobre a existência do caminho fechado. A questão de determiná-lo foi resolvida cem anos mais tarde por [Heierholzer 1873].

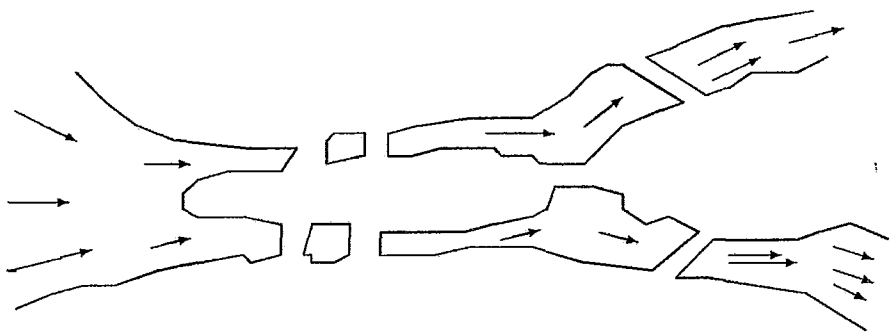


Figura VI.1: Vista das sete pontes de Königsberg na ilha de Kneiphof sobre o rio Pregel.

Um outro problema muito próximo ao problema das pontes de Konigsberg é o Problema do Carteiro Chinês (PCC), que foi introduzido por Kwan Mei-Ko, um matemático da Universidade Normal de Shangtun. Quando de sua passagem como funcionário dos correios durante a revolução cultural chinesa, Kwan além de se preocupar com o caminho fechado que passasse por todas as ruas de trabalho, levou em consideração a necessidade de percorrê-las a todas pelo menos uma vez com o menor comprimento. Kwan definiu assim o problema: *Um carteiro tem que cobrir seu local de trabalho antes de retornar ao posto. O problema é encontrar a menor distância de percurso para o carteiro.*

O termo percurso em arcos abre muito a interpretações duvidosas, principalmente em virtude de sua relação com a teoria dos grafos. Para dirimir este problema conceitual neste texto, como foi definido na seção I.1, a definição de que $G = (V, L)$ é um grafo, ou seja, é um conjunto formado por uma dupla de conjuntos, sendo V o conjunto de vértices (nodos ou pontos) e L o conjunto das ligações entre vértices as quais são formadas por uma nova dupla de conjuntos $L = (E, A)$, onde E é o conjunto de tipos não-orientados (elos) e A é o conjunto de tipos orientados (arcos). Em geral a literatura chama de arcos como o conjunto das ligações, porisso a grande confusão.

É importante definir o Problema de Percursos em Arcos (PPA), como um problema especial da classe de Problemas Gerais de Roteamento (PGR), [Orloff 74] e [Male et al 77]. Os Problemas Gerais de Roteamento ou Percurso, consideram que um subconjunto de vértices $V^s \subseteq V$ e um subconjunto de ligações $L^s \subseteq L$ serão atendidos a partir de um vértice origem, por um circuito que percorre todos os elementos de V^s e L^s com custo total mínimo.

Alguns dos casos especiais do PGR são:

$L^s = \emptyset$ e $V^s = V$	PGR $\equiv$ PCV
$V^s = \emptyset$ e $L^s = L$	PGR $\equiv$ PCC
$V^s \subset V$ e $L^s = \emptyset$	PGR $\equiv$ Problema do Caixeiro Seletivo (PCS)
$V^s = \emptyset$ e $L^s \subset L$	PGR $\equiv$ Problema do Carteiro Rural (PCR)
$V^s \subset V$ e $L^s \subset L$	PGR $\equiv$ (Puro)

As aplicações de percurso em arcos podem ser encontradas em um grande número de situações reais (práticas), algumas que se destacam são relatadas a seguir.

Aplicações do Problema do Carteiro Chinês, [Eiselt et al 95i]

1. **Teste Topológico de Sistemas Computacionais em Diferentes Níveis**, Nestes testes os vértices são tomados como processadores, interruptores e registradores, dependendo do nível específico cujo teste está

para ser conduzido, e os arcos representam o fluxo de dados ou linha de dados entre vértices. Um circuito euleriano pode ser usado para enviar um pacote teste que atravessa todos os arcos e vértices no menor tempo, [Malek et al 89].

2. **Determinação de Estados Mínimos de Energia em Vidros Spin,**
   Um sistema de vidros *spin* pode ser modelado por um grafo não orientado no qual cada vértice corresponde a um *spin* e os valores dos elos indicam as interseções entre *spins* (o custo do elo representa o nível de interação positiva e negativa entre os *spins* que ele liga).

   O estado de energia mínimo consiste na atribuição de orientações +1 ou -1 a *spins* com o objetivo de minimizar a energia total do sistema, dado pelo oposto da soma de todos os elos do produto da orientação dos *spins* e o custo das orientações.

   A solução ótima deste problema pode ser obtida encontrando-se o custo mínimo de um grafo, particionando os *spins* de orientação +1 dos *spins* -1.

   Para sistemas planares de vidros *spin*, esse corte de peso mínimo pode ser obtido pela resolução de um problema em arcos sobre um grafo dual planar do grafo original que descreve o sistema. Neste grafo dual, o custo de atribuição a um elo é o valor absoluto do custo de interação do único elo atravessado no grafo original percorrido. A solução corresponde ao ciclo de custo mínimo que cobre os elos duais associados a cada elo de peso negativo no grafo original, [Barahona 90].

3. **Problema de Minimização de Vias no Projeto de Circuitos VLSI,**
   Neste problema considera-se que os fios que conectam os componentes de um *chip* devem ser atribuidos a uma das duas camadas dos *chips*. Os fios não podem cortar uns aos outros sempre quando pertencem à mesma camada, mas podem ser conectados à outra camada através de vias (conecções entre camadas). O objetivo é minimizar o número de vias.

   Para resolver este problema, constrói-se inicialmente uma representação de um grafo planar do *chip* cujos vértices correspondem aos pontos finais dos fios e às travessias, e os elos correspondem a seções de fios entre esses pontos. A solução é obtida pela determinação do menor número de elos que devem ser quebrados em dois pedaços de modo a gerar um grafo bipartido (cada ocorrência correspondente a uma via).

Aplicações do Problema do Carteiro Rural

1. **Coleta de Lixo**
   Nesta aplicação, uma grande soma em dinheiro é gasto anualmente em todas as cidades mundiais com o problema da remoção do lixo urbano.

Cerca de 70% dos recursos alocados à atividade são destinados à remoção pura e simplesmente (caminhões, rodagem e turma de coleta).

Em minha experiência de campo, basicamente nas cidades de Fortaleza e Rio de Janeiro (zona sul), identifiquei somas de produção de lixo domiciliar na ordem de 1200t (1995) e 620t (1990) em média por dia para Fortaleza e a Zona Sul do Rio de Janeiro, respectivamente. O custo da tonelada recolhida na prefeitura do Rio de Janeiro em 1991 ocilava em torno de US$ 69,00 enquanto em Fortaleza o custo era de US$ 40,00[1]. Milhões de dólares anuais são gastos com esta atividade no Brasil e no mundo como pode-se perceber. A coleta de lixo possui, pois, um papel destacado nos setores municipais.

O problema da coleta de lixo domiciliar, considera em geral que um município é dividido em várias regiões de coleta e cada uma destas regiões são particionadas em sub-regiões as quais são dimensionadas para serem atendidas por um veículo compactador cujo tipo (tecnologia) depende da verticalidade da sub-região que será atendida.

O serviço de coleta é executado por um caminhão percorrendo um conjunto de segmentos de rua, em um número de viagens entre a sub-região e o local de despejo (estações de transferências, aterros sanitários ou lixões) que depende fortemente do tempo de serviço da tripulação, apesar da força contratual designar que a sub-região deve ser completamente atendida no seus dias determinados. Cada segmento de rua pode conter pontos especiais de demanda com um peso relativamente alto em relação aos demais segmentos, caracterizando pois um problema Geral de Roteamento. A rede viária é em geral assimétrica (maioria das ruas de mão única) nas sub-regiões mais urbanizadas possuindo fortes restrições de tráfego, já as sub-regiões da periferia têm grandes características de simetria (maioria das ruas de mão-dupla). Em complexos habitacionais (favelas, grandes condomínios, etc.), o lixo é coletado em grandes caixas pelos compactadores em pontos estratégicos para que se facilite o acesso dos veículos de remoção.

A frota é em geral heterogênea, e o seu dimensionamento é um dos fatores cruciais nos custos de desenvolvimento dos trabalhos de remoção.

Alguns problemas surgem diretamente do problema da coleta de lixo domiciliar, [Bodin & Beltrami 74], [Levy & Bodin 89], [Bodin et al 89], [Negreiros 90], [Negreiros et al 92a], [Negreiros et al 92b], [Negreiros & Maculan 92a], [Negreiros & Maculan 92b], [Negreiros et al 92], [Eiselt et al 95ii]. Os mais importantes que foram considerados pela literatura são:

(a) Dimensionamento das regiões de coleta considerando uma frota dis-

[1]Informações obtidas das planilhas Indicadoras de Produção e Custo (PCC), COMLURB-RJ, Jan-Mar/1991 e dos indicadores trimestrais (3o. Sem. 1995), EMLURB-Fortaleza.

ponível, os locais de despejo e o custo global de atendimento;

(b) Escalonamento das regiões de coleta dentro de um horizonte com periodicidade de atendimento regular;

(c) Dimensionamento da tripulação para dada sub-região;

(d) Roteirização dos percursos, com o objetivo de reduzir o esforço e o custo de transporte no serviço diário;

(e) Acompanhamento da frota para o dimensionamento da reserva técnica ideal de veículos;

(f) Escalonamento do pessoal da coleta;

(g) Escalonamento da rotatividade dos caminhões com o objetivo de reduzir o desgaste e cumprir os procedimentos ligados à manutenção, dentre outros.

Quando o problema trata simplesmente do percurso de um veículo em uma sub-região fixa, seu objetivo está ligado à minimização do custo total de recolhimento do lixo na sub-região, sujeito à capacidade do veículo e ao tempo de trabalho da tripulação. Sabe-se no entanto que o veículo parte de uma garagem e a ela retorna, porém toda vez que sua carga é preenchida, o veículo tem que ir a um local de despejo, até que toda a sub-região seja atendida, figura VI.2.

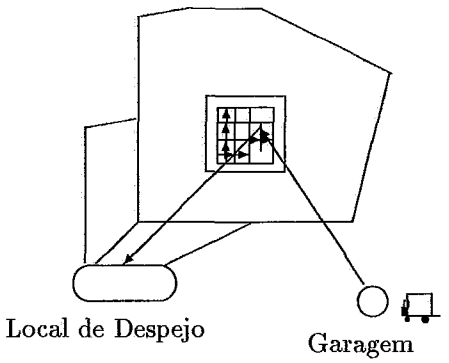


Figura VI.2: Situação de coleta definida para uma dada sub-região.

Uma questão importante é sobre a forma de prever a produção do lixo. Quanto ele pesa? Como se encontra roteiros determinísticos que conseguem atender a demanda de uma sub-região em virtude das variações de produção diária? Em [Negreiros 90] e [Negreiros et al 92a] mostra-se que o lixo tem uma série histórica com um índice de correlação muito alto entre os elementos da produção (Peso, viagem, quilometragem, tempo de serviço, dias de serviço). Isto o torna previsível nos casos de pico máximo e nos médios para os diversos dias de coleta. No caso da variação entre o primeiro e demais dias ser muito alta, recomenda-se planejar um serviço de coleta com rotas diárias, porém dependerá muito desta variação. Na

prática, o que se recomenda é que os roteiros se mantenham firmes por mais tempo possível, fato que facilita sobremaneira a vida dos motoristas.

Os resultados obtidos com a aplicação de heurísticas neste contexto, variam regularmente dependendo do tamanho dos circuitos. No Rio de Janeiro, sub-regiões muito curtas com percursos entre 15 e 20km, porém de produção elevada 28t, atingiu-se um aumento em eficiência (kg/km-h) em torno de 4%, já em regiões maiores entre 30 e 35km e 20 a 25t chegou-se até a 50% de aumento na eficiência, nos custos em distância porém a economia foi de 10% em média. No caso de Fortaleza em uma subregião com 20km de percurso e produção de 16t, atingiu-se marcas que chegaram a 25% na redução da distância total e um aumento em 88% na eficiência da coleta, [Negreiros 90] e [Negreiros et al 92a]. A economia projetada para o Rio de Janeiro era em torno de US$ 250mil anuais, enquanto que em Fortaleza não se chegou a nenhuma perspectiva.

2. **Varredores de Rua**

Neste problema veículos especiais com vassouras circulares são escalonados para percorrer um conjunto de ruas de uma cidade, de tal modo que atendam em seu percurso às restrições de tráfego como sentidos, paradas obrigatórias e horários de trabalho. Assim, o problema está relacionado à seleção destas ruas que incluem a cada rota uma carga de trabalho que devem ser balanceadas e ruas que devem ser atendidas com certa frequência.

A curiosidade deste problema está principalmente no nível de exigência do sistema de varrição municipal. [Bodin & Krush 78] e [Bodin & Krush 79] tiveram que tratar o problema da formulação com restrições de tráfego, já [Eglese & Murdock 91] discutiram sobre a falta da necessidade de se colocar as restrições de movimentação uma vez que na cidade onde foi aplicado o modelo havia poucas ruas de mão única e estas seriam percorridas pelo veículo varrendo na conta-mão. A grande discussão está no respeito ou não às normas, em contrapartida ao aumento excessivo na quantidade de dados necessárias pelo sistema de informações que tratará as instâncias.

De minha experiência em campo, quando já evidenciei o perigo de se trafegar na contra-mão com um veículo de limpeza urbana, não tenho dúvidas que o problema deve ser encarado dentro de sua realidade tirando o máximo proveito dos limites. Os limites, pois, devem ser rigorosamente respeitados.

3. **Entrega de Correspondências**

A maioria dos estudos relacionados à entrega de correspondências foi feito por [Levy & Bodin 88], [Levy & Bodin 89] nos Estados Unidos e [Roy & Rousseau 89] no Canadá. No caso dos EUA, o problema se encontra na partição dos segmentos de rua em grupos (*clusters*), para lo-

calizar depósitos usados como pontos iniciais dos roteiros e determinar rotas para cada grupo. Os grupos devem ser balanceados e correspondem ao volume máximo de correspondência que deve ser entregue ao mesmo tempo.

Todos os dias um carteiro dirige um veículo a partir da sua agência dos correios para um local de estacionamento, chamado de *depósito*. Desprezando-se o tempo de condução até o depósito, o carteiro deixa as correspondências a um dos grupos adjacentes ao depósito, retorna ao carro e entrega as cartas a um outro grupo adjacente ao depósito e assim por diante até que todos os grupos atribuídos àquele depósito sejam atendidos. O entregador então dirige-se a um outro depósito seguindo o mesmo procedimento. Isto se repete até que o dia de trabalho seja completado. O grande mérito desta aplicação está num problema de localização dos depósitos, que está um pouco fora do Problema de Percurso em Arcos.

[Roy & Rousseau 89] também viram seu problema como um problema de localização e roteamento, exceto que neste problema há um depósito diferente associado ao início de cada rota do carteiro. Consideraram que o custo de localizar um depósito é igual ao custo de viagem entre ele e a agência. Todas as rotas, incluindo o tempo de viagem aos depósitos e de retorno são limitadas à duração do dia de trabalho (horas), mas não pela capacidade uma vez que caixas de controle estão localizadas de forma apropriada ao longo de cada rota.

4. **Percurso do Fumacê - Prevenção contra o Dengue**

O Dengue e o Dengue Hemorrágico são doenças provocadas por mosquitos que vêm tendo um dos crescimentos mais impressionantes e rápidos no que concerne às infecções por arbovirus no mundo. Mais de 100 países, entre tropicais e sub-tropicais, já tiveram alguma experiência com esta doença. Há 10 milhões de casos por ano, dos quais 95% são de crianças. A taxa de pessoas atacadas é de 6700 a cada 100mil expostos, cerca de 500mil casos de internação são registrados ao ano, onde mil casos de morte ao ano são registrados. A epidemia por dengue tem sérias consequências socio-econômicas, e é extremamente caro e difícil o seu controle, [WHA 96].

Dentre os fatores que contribuiram para o seu crescimento, considera-se principalmente a rápida expansão das populações urbanas, fazendo com que mais e mais pessoas tenham contato com mosquitos domésticos ou peri-domésticos, especialmente o *Aedes Aegypti*, cuja população continua a crescer. A ausência de fontes de água de boa qualidade e segura, cisternas de armazenamento d' água bem feitas, posicionamento de esgoto em local seguro e fechado, oferece condições para o desenvolvimento e proliferação dos mosquitos. Devido à Dengue ser causada por um de cinco

virus distintos (soros), os esforços para o desenvolvimento de uma vacina efetiva ainda não tiveram sucesso.

Em 1993 a WHA (World Health Assembly) enfatizou que a prevenção e o controle do Dengue é de alta prioridade. Ela definiu um conjunto de estratégias que são adaptáveis às situações regionais e nacionais. Com o objetivo de atentar claramente para as resoluções do WHA, uma atenção especial deve ser dada ao:

- Desenvolvimento de estratégias detalhadas com o objetivo de reduzir as internações e a mortalidade as quais podem ser empregadas em qualquer lugar;
- Operacionalização dos melhores métodos de controle de propagação do mosquito.
- Formulação de programas para vigilância da doença e da propagação do mosquito.
- Aumento do suporte político a nível nacional.

Considerando os aspectos de controle de proliferação do mosquito, o trabalho no Brasil vem sendo desenvolvido pela Fundação Nacional de Saúde - FNS, principalmente nos períodos chuvosos no nordeste e no sul, quando da maior incidência do aumento da população de mosquitos domésticos.

A grande incidência de transmissão no Brasil, deve-se ao mosquito *Aedis Aegypti* adulto ou de suas larvas, que são encontrados em águas limpas (claras) ideal para a sua reprodução. O mosquito possui um ciclo de vida médio de sete dias, e o elemento infectante é a fêmea.

O combate à doença vem sendo feito através de campanhas de prevenção pela eliminação de focos tais como depósitos de água limpa colocados em qualquer lugar dentro e fora das residências. Estando os focos da doença identificados pelas unidades de fiscalização, o sistema de combate aciona *guardas* que atuam de residência em residência, unidades comerciais, cemitérios, etc. colocando inseticidas em pó nos depósitos suspeitos.

Uma outra medida de combate é executada através da colocação de veículos (*Fumacê*) equipados com bombas de pulverização tipo LECO-HD1800, que pulverizam o ar próximo à fachada das residências com inseticidas organo-fosforados. Geralmente a pulverização é realizada sempre que os indicadores de casos por bairro das cidades atingem 2%. O serviço é feito sempre nos horários de calmaria de vento (velocidade ≤ 6 km/h) 5:00 às 7:00 e de 17:00 às 19:00. Este procedimento se repete em intervalos fixos de no máximo cinco dias por cinco a seis vezes (ciclo). Após isto, uma amostragem é feita na região para avaliar a permanência ou não da doença.

Em geral o *Fumacê* viaja a uma velocidade média de 16 km/h com a bomba em funcionamento a uma vazão entre 90-127 ml/min. O percurso

é feito com o bico aspersor virado para a parte frontal das residências e o mais próximo possível do meio fio, atravessando sempre os quarteirões nos dois lados, ver figura VI.3.

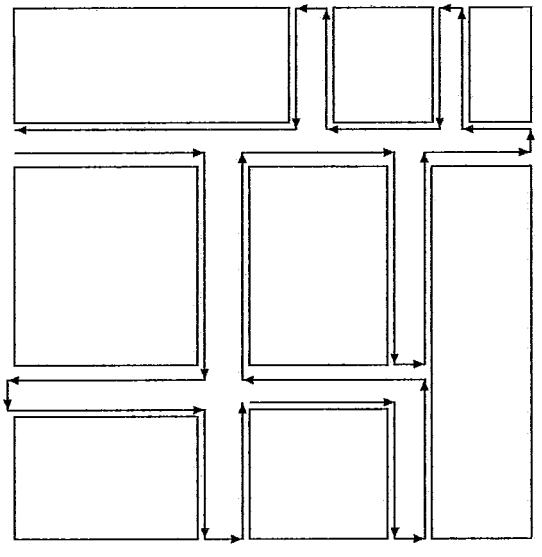


Figura VI.3: Exemplo do percurso de um veículo *Fumacê*.

A característica do percurso do *Fumacê* é muito parecida com o entregador de correspondências, porém em avenidas onde o fluxo de tráfego está definido é importante que seja feito conforme sua direção. Em ruas secundárias de mão única é permitido que o tráfego seja realizado com o veículo na contra-mão porém com sinalização de alerta.

O sistema de controle do Dengue possui três bases no Brasil, Eusébio (Fortaleza), Uberaba (Minas Gerais) e Feira de Santana (Bahia), sendo a SUCAM o orgão responsável pela operacionalização do serviço.

Os maiores focos de 1996 foram registrados em Salvador e em São Luiz. Em 1994 foram registrados aproximadamente 30 mil casos com estimativa real de uma epidemia, que atacou cerca de 825mil pessoas somente em Fortaleza, fato que compromete toda a população uma vez que em caso de nova manifestação da doença espera-se que uma forma do arbovirus com maior grau de infecção se manifeste. Como prevenção, a FNS vem gastando aproximadamente US$ 576 mil por ciclo somente com o controle através dos guardas de epidemia [2].

5. **Percurso de** *Tapa Valas*

[2]Todas as informações aqui incluidas foram gentilmente fornecidas em caráter extra-oficial pelo Dr. Humberto Melo Mendonça (Gerente de Controle do Dengue e Febre Amarela do SOE-FNS/Ce)

Em regiões de periferia de grandes cidades brasileiras, o problema do saneamento básico e de infra-estrutura são evidentes. Em geral a concentração destes serviços é feita por quadras, através da execução de obras de infraestrutura em segmento de ruas especificados pelas secretarias de obras dos municípios. Após completados os serviços de colocação da infraestrutura, um conjunto de empresas de construção vêm com seus caminhões e turmas, para fazer o fechamento e reurbanização dos locais onde foram instalados os benefícios.

As empresas que executam estas operações, geralmente têm que planejar a sua capacidade de serviço e seus recursos com o objetivo de atender a todas as ordens contratadas com o município. Estas ordens em geral têm prioridades de execução, porém são realizadas de forma segmentada simultaneamente em diversos pontos da cidade. Há serviços que demoram desde uma fração de dia a uma semana. Como geralmente um sequenciamento de equipamento é necessário (unidade de asfalto, unidade de planificação do terreno, etc.) estes equipamentos podem estar circulando na rede de serviço toda vez que seu trabalho tenha-se completado.

O problema é minimizar os percursos da frota sobre regiões de serviço, partindo de um depósito central, conhecendo-se a taxa de execução das operações por segmento de rua, e escalonar a frota que atenderá os contratos em um dado período. A grande característica deste problema é o seu dinamismo.

6. **Percurso de Leituristas de Energia e Gás**

No percurso de leituristas de energia, o objetivo é encontrar m rotas de duração limitada que cobrem uma certa área, iniciando de um depósito central. O objetivo é minimizar m. [Stern & Dror 79] aplicaram seu estudo na cidade de Beesheva, Israel, obtendo uma economia em torno de 40% sobre o custo do atual sistema. Um problema similar foi tratado por [Wunderlich et al 92] para a Companhia de Gás do Sul da Califórnia, resultando numa economia anual em torno dos US$ 0,9 milhões.

7. **Percurso de Ônibus Escolares**

Poucas aplicações de roteamento em arcos são tão intrigantes quanto a de percurso de ônibus escolares. Aqui, os dois objetivos de minimizar o número de ônibus e a quilometragem são sempre otimizados de um modo hierárquico à medida que custos fixos tendem a predominar. Além das restrições de tráfego usuais, algumas restrições devem ser satisfeitas, tais como janelas de tempo, número mínimo e máximo de passageiros no ônibus em um dado momento, a variedade de estudantes (escolas, turmas, etc.), e o tempo máximo que qualquer aluno passa no ônibus, [Eiselt et al 95ii]. Nas implementações mais recentes, [Desrosiers et al 86] usaram um algoritmo de geração de colunas para escalonar 60 escolas e 20mil estudantes.

8. **Remoção de Neve**
   Os removedores de neve são veículos especiais equipados com produtos químicos que dissolvem a neve, com o objetivo de liberar as vias de uma cidade que sofre nos períodos de inverno com tempestades de neve. O problema tem restrições especiais de movimentação em relação ao material químico e certas prioridades de travessia a determinadas ruas (deve-se passar numa determinada rua primeiro que em outra). A literatura também reporta para este caso os cuidados que devem ser tomados em virtude dos momentos de contingência como num caso de nevasca, [Alprin 75], [Cook & Alprin 76], [Haslam & Write 91], [Gelinas 92].

Formalizando os Problemas de Percurso em Arcos, seja $G = (V, L)$ um grafo conexo sem *loops*, onde $V = \{v_1, v_2, \ldots, v_n\}$ é o conjunto de vértices (ou conjunto de nós), e $L = \{(v_i, v_j) \ : \ v_i, v_j \ \in V \ e \ i \neq j\}$ é o conjunto de ligações. (Em alguns problemas G é um multigrafo, ou seja, algumas ligações de G podem ser paralelas para um mesmo par de vértices). A cada ligação (v_i, v_j) está associado um custo não negativo, que pode ser encarado como a distância, tempo ou comprimento de c_{ij}.

A matriz C sendo simétrica, é comum associar um conjunto de elos E (ligações não orientadas) a cada ligação, ou par de vértices do conjunto L, melhor dizendo $L = E$. Sendo C assimétrica, é possível que todas as ligações sejam orientadas (grafo orientado) assim como apenas uma parte delas, sendo o restante não orientado (grafo misto). Nas ligações orientadas dizemos que se forma um conjunto de arcos A, onde $A \subseteq L$. Portanto, dependendo da formação do conjunto L o problema de Percurso em Arcos será caracterizado. É possível que uma matriz C satisfaça à desigualdade triangular se e só se $c_{ik} + c_{kj} \ \leq \ c_{ij}, \ \forall \ i, j, k$.

A seguir discute-se basicamente o desenvolvimento teórico de quatro correntes de roteamento em arcos, o problema do Carteiro Chinês (PCC) em suas várias versões da literatura (casos não-orientado, orientado, misto, íngreme e hierárquico), o Problema do Carteiro Rural (PCR) e suas versões (não-orientado, orientado e o Problema *Stacker Crane*), e o Problema de Percurso (Roteamento) em Arcos com Demanda (PRAD).

VI.1 O Problema do Carteiro Chinês

Como foi visto, o PCC trata da travessia de custo mínimo de todas as ligações entre vértices de um grafo valorado a partir de um dado vértice e a ele retornando.

Em geral, os grafos têm que ser percorridos de modo que suas ligações precisam ser replicadas, [Euler 1736]. Para que o PCC seja melhor compreendido,

estende-se o teorema de Euler para grafos orientados e mistos com a *Propriedade de Unicursalidade.*

Propriedade de Unicursalidade

Seja G um grafo conexo. G é dito ser *unicursal* ou *euleriano* se existe um caminho fechado em G contendo cada aresta apenas uma vez e cada vértice pelo menos uma vez. As condições necessárias e suficientes para que um grafo seja conexo e euleriano são dadas como segue.

1. Se G é não orientado, todo vértice deve ter grau par, ou seja, um número par de elos incidentes - *Teorema de Euler.*

2. Se G é orientado, o número de arcos entrando e saindo de cada vértice é igual - *Teorema de Ford & Fulkerson*, [Ford & Fulkerson 62];

3. Se G é misto, todo vértice em S deve conter um número par de arcos orientados a ele ligados; além disto, para todo conjunto $S \subseteq V$, a diferença entre o número de arcos de S para $S \setminus V$ e o número de arcos de $V \setminus S$ para S deve ser menor do que ou igual ao número de elos ligando S e $V \setminus S$ - *Condições de Balanceamento*, [Nobert & Picard 91], figura VI.4.

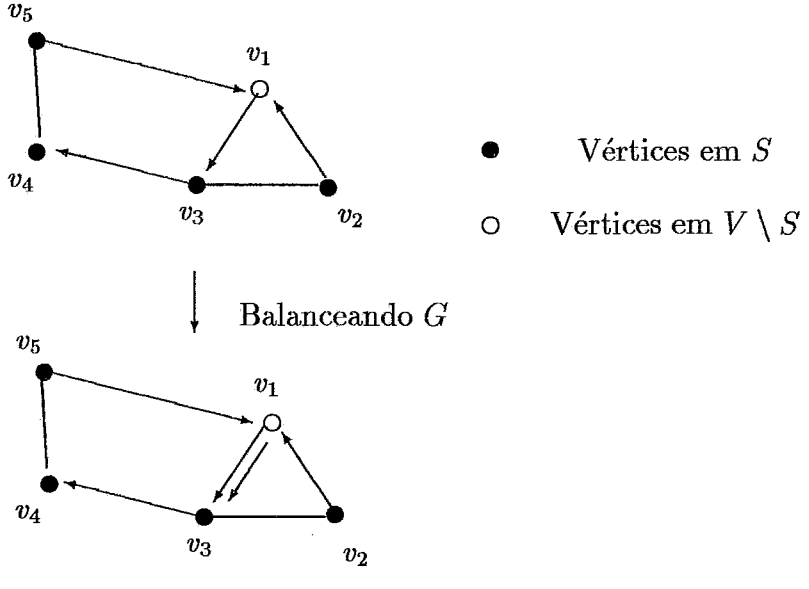


Figura VI.4: Satisfação das Condições de Balanceamento em um grafo misto.

A figura VI.4 mostra que no grafo G tomando o conjunto $S = \{v_1\}$ e o conjunto $V \setminus S = \{v_2, v_3, v_4, v_5\}$, tem-se no grafo superior que o número de arcos de S para $V \setminus S$ é 2 (A) e o número de arcos de $V \setminus S$ para S é 1 (B), e o número de elos

ligando S e $V \setminus S$ é 0 (C). Portanto a diferença entre (A)-(B)>(C), não verificando a condição de balanceamento. Já para a situação do grafo G após o balanceamento, nota-se que (A)-(B)=(C), verificando pois esta condição.

Com isso, pode-se considerar diretamente que os métodos de resolução do PCC contêm em geral dois estágios distintos. O primeiro trata de encontrar um grafo aumentante G_a de G de custo mínimo, ou seja, um novo grafo cujo novo conjunto de ligações tornará G um grafo euleriano. E o segundo trata do percurso do carteiro a partir de um dado vértice passando por todas as ligações de G_a e retornando ao vértice de partida.

VI.1.1 O PCC Não Orientado

Quando $G = (V, E)$ é não orientado, o problema de encontrar o grafo G_a pode ser resolvido através de um algoritmo de *1-Matching* não bipartido valorado sobre os pares de vértices de grau ímpar de G, [Edmonds & Johnson 73], figura VI.5.

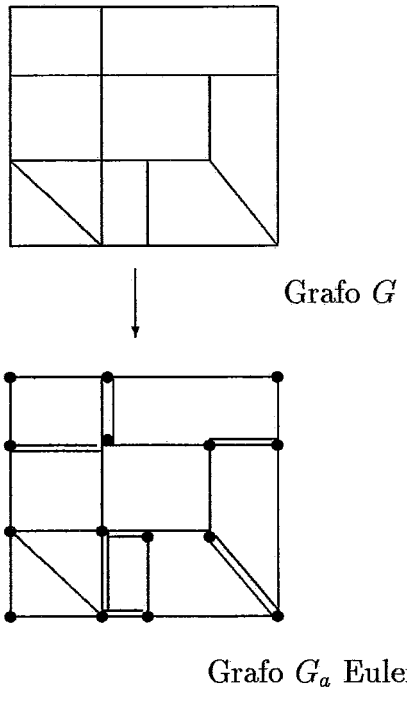


Figura VI.5: Um grafo aumentante do PCC.

Formulando-se o problema matematicamente, define-se x_{ij} como o número de réplicas de (v_i, v_j), $i < j$ necessárias para aumentar G. Seja $\delta(i)$ o conjunto de elos incidentes a v_i, e seja $T \subseteq V$ como o conjunto de vértices de grau ímpar de V. Um modelo para o PCC não-orientado pode ser escrito como segue:

$$\text{(PCC Não Orientado-1) Minimizar} \sum_{(v_i,v_j)\in E} c_{ij}x_{ij} \tag{VI.1}$$

sujeito a,

$$\sum_{(v_i,v_j)\in\delta(i)} x_{ij} \equiv \begin{cases} 1 \pmod 2 & \text{Se } v_i \in T \\ 0 \pmod 2 & \text{Se } v_i \in V \setminus T \end{cases} \tag{VI.2}$$

$$x_{ij} \in \{0,1\}, \quad \forall\, (v_i, v_j) \in E \tag{VI.3}$$

Como mostrou [Barahona 90], a formulação acima independe da densidade de ligações de G[3]. Uma formulação equivalente que explora a esparsidade de G foi proposta por [Edmonds & Johnson 73]. Nela define-se qualquer subconjunto próprio não vazio de V como o conjunto $E(S) = \{(v_i, v_j) : v_i \in S,\ v_j \in V \setminus S \text{ ou } v_i \in V \setminus S,\ v_j \in S\}$.

$$\text{(PCC Não Orientado-2) Minimizar} \sum_{(v_i,v_j)\in E} c_{ij}x_{ij} \tag{VI.4}$$

sujeito a,

$$\sum_{(v_i,v_j)\in E} x_{ij} \geq 1 \ (S \subset V,\ S \text{ é ímpar}) \tag{VI.5}$$

$$x_{ij} \in \mathcal{Z}_+ \quad \forall\, (v_i, v_j) \in E \tag{VI.6}$$

Nesta formulação a grande diferença está nas equações VI.5 que se referem a desigualdades de *blossons*[4]. Elas estão definidas para todo par de vértices ímpares de S, ou seja, para cada subconjunto próprio de V contendo um número de vértices com grau ímpar. Um resultado interessante proposto por [Edmonds & Johnson 73] é que o poliedro formado pelas equações VI.5 e VI.6 é igual à envoltória convexa das soluções do problema (PCC Não Orientado-2).

Para formar o ciclo euleriano no grafo G_a, pode-se inicialmente aplicar o algoritmo de achar o percurso em um grafo euleriano proposto por [Edmonds & Johnson 73].

Algoritmo **C-Euler**

Passo 1: Tomar s (Vértice de partida), escolher um vértice x entre aqueles ligados a s. Armazenar (s, x) como elo visitado;

[3]Densidade de um grafo, é o número médio de ligações que saem dos vértices de G.

[4]*Blossons* são circuitos com $2k+1$ vértices os quais têm k elos emparelhados em um *matching* fixo, [Papadimitriou & Steiglitz 82]

Passo 2: Continuar tomando os vértices x' ligados a x dos elos não visitados, até que $x' \equiv s$. Armazenar (x, x') como elo visitado.

Passo 3: Caso haja algum elo ainda a visitar, ir para o passo 2, senão terminar.

Os algoritmos de resolução deste problema são da ordem de $O(n^3)$ para a resolução do grafo aumentante, n é o número de vértices de grau ímpar em G, e de $O(m)$ para a resolução do algoritmo C-Euler, onde m é o número de elos do grafo aumentante, [Eiselt et al 95i].

Algoritmos como o C-Euler, não se preocupam na verdade com a forma do trajeto do carteiro. Algumas vezes, nas aplicações reais, é necessário que a rota construída seja ponderada por certos critérios de continuidade. Elementos importantes como se manter fora de serviço, dobrar à esquerda, fazer zig-zags e retornos em "U" podem significar uma perda razoável de tempo num serviço doméstico diário. Recursos para dar um encaminhamento a um percurso mantendo-o em serviço, fazendo poucas travessias de rua e fazendo-o realizar poucos retornos em "U" podem ser vistos em [Bodin & Krush 79] e [McBridge 82], e serão discutidos melhor no próximo capítulo.

Um conjunto de variações do PCC não orientado podem ser encontradas em [Frederickson et al 78]. Entre as variações mencionadas, encontra-se o problema do m-Carteiro Chinês, onde o objetivo é minimizar o comprimento das maiores rotas dos carteiros. Este problema é uma extensão da idéia do m-PCV, porém relacionada à formação equilibrada de um certo número de elos a visitar por cada m-carteiro. Os autores mostraram que este problema é do tipo $\mathcal{NP}$-Árduo através da redução ao problema da k-Partição. Foi apresentada uma heurística com complexidade de $O(2 - 1/m)$.

Outro problema relacionado ao PCC pode ser o problema da cobertura de elos de um grafo através de ciclos não orientados. O objetivo é a cobertura de todos os elos de um grafo simétrico por ciclos simples tal que o comprimento total de cobertura é minimizado. Este problema surge no projeto de sistemas de irrigação, e na análise de circuitos elétricos, [Cross 36] e [Itai & Rodeh 78]. [Itai et al 81] mostraram que a complexidade do problema de determinação da cobertura de comprimento mínimo não é conhecida, mas conjecturaram ser $\mathcal{NP}$-Árduo. Eles também mostraram que a minimização do comprimento de cobertura e a solução do PCC não são equivalentes para um grafo qualquer, porém [Kesel'Man 87] provou que para grafos planares os dois problemas são equivalentes.

VI.1.2 O PCC Orientado

No caso orientado considera-se $G = (V, A)$ um grafo onde as ligações são formadas por $A = \{< v_i, v_j >: i \neq j \ e \ v_i, v_j \in V\}$ o PCC pode ser resolvido a partir da determinação do grafo euleriano de G através do fluxo de custo mínimo onde todo arco de G recebe como limite inferior uma unidade de fluxo.

Formalmente, seja I o conjunto dos vértices v_i os quais o número de arcos entrando é maior que o número de arcos saindo ligados a partir de um (novo) vértice f (fonte), e seja J o conjunto dos vértices v_j onde o número de arcos saindo excede o número de arcos entrando ligados na direção de um (novo) vértice s (sumidouro).

Supondo c_{ij} o comprimento do arco $< v_i, v_j >$. Tomando-se $d^+(i)$ e $d^-(i)$ como o número de arcos saindo e entrando no vértice v_i, respectivamente. Ao contrário da condição de que G precisa ser conexo para existir uma solução do Carteiro Chinês, esta condição não basta para o caso orientado, nela G precisa ser fortemente conexo para que haja um grafo euleriano orientado. A formulação do PCC orientado é descrita a seguir.

$$\textbf{(PCC Orientado) Minimizar} \sum_{v_i \ in I} \sum_{v_j \in J} c_{ij} x_{ij} \tag{VI.7}$$

sujeito a,

$$\sum_{v_j \in J} x_{ij} = |d^+(i) - d^-(i)| \ (v_i \in I) \tag{VI.8}$$

$$\sum_{v_i \in I} x_{ij} = |d^+(j) - d^-(j)| \ (v_j \in J) \tag{VI.9}$$

$$x_{ij} \geq 0 \ (v_i \in I, \ v_j \in J) \tag{VI.10}$$

Uma outra abordagem de solução do PCC orientado está relacionada ao problema de transporte, [Edmonds & Johnson 73], [Bodin & Beltrami 74], e [Yaxiong & Yongchang 88].

Um exemplo do procedimento de solução acima proposto pode ser visto na figuras VI.6, VI.7, VI.8, onde os valores sobre os arcos referem-se a seus custos, [Negreiros 90]. Na figura VI.7, tem-se o grafo expandido G^k, onde a fonte f incide sobre todo vértice cuja soma de arcos entrando é maior que saindo. Já todo vértice que possui um número de arcos saindo maior do que entrando possuem um arco que incide sobre o vértice sumidouro s. Colocando-se de forma apropriada os limites de fluxo por arco no grafo expandido, aplica-se um algoritmo de fluxo em redes onde o resultado designará a quantidade de fluxo circulante entre cada arco.

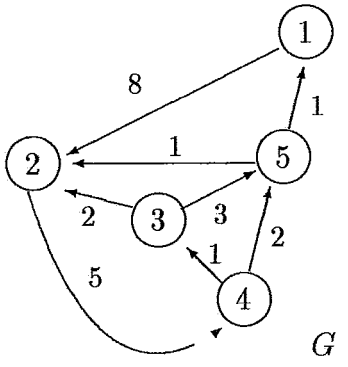


Figura VI.6: Grafo original orientado.

Estas unidades de fluxo adicionais significam o número de réplicas do arco, como pode ser visto no multigrafo da figura VI.8.

Uma vez que G_a é euleriano, apesar de orientado, o procedimento C-Euler se aplica para encontrar o caminho euleriano.

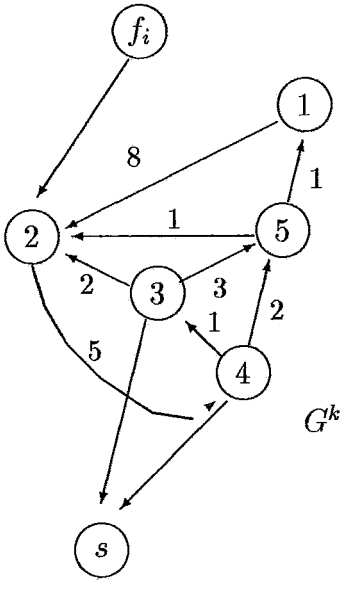
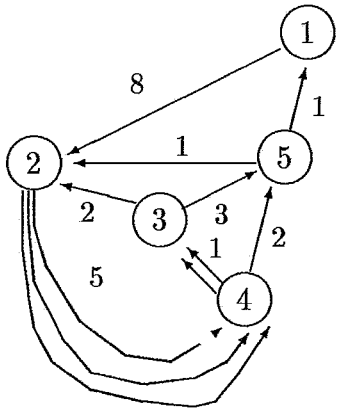


Figura VI.7: Grafo orientado preparado para o Problema de Fluxo de Custo Mínimo.



G Solução

Figura VI.8: Grafo aumentado Euleriano resultante.

VI.1.3 O PCC Misto

O PCC misto é um problema da classe de problemas $\mathcal{NP}$-Árduos. Contém uma generalização dos grafos não-orientados e orientados, onde existem um conjunto de elos e arcos não vazios, [Lenstra & Rinooy Kan 81].

Seja $G = (V, L)$ um multigrafo fortemente conexo, onde $L = (E, A)$, sendo $E = \{(v_i, v_j) : v_i, v_j \in V\} \neq \emptyset$ e $A = \{< v_s, v_k >: v_s, v_k \in V\} \neq \emptyset$, e G_a o grafo aumentado de G sendo G_a euleriano. G é par se o número de elos e arcos incidentes a todo vértice é par. G é simétrico se para cada vértice o número de arcos entrando é igual ao número de arcos saindo. Um grafo é balanceado, se as condições de balanceamento da unicursalidade são satisfeitas.

Se G é par e simétrico, então ele é balanceado, porém a simetria não é uma condição suficiente à unicursalidade. Se já se sabe que G é euleriano, pode-se ter o problema de determinação de um circuito euleriano em G. Isto pode ser atingido em três fases:

1. Atribuir direções a alguns elos de forma que G seja simétrico;
2. Atribuir direções aos elos restantes;
3. Determinar a travessia atual de G.

[Ford & Fulkerson 62] propuseram o seguinte procedimento para transformar um grafo misto em um grafo simétrico.

Passo 1: Trocar cada elo de G com um par de arcos orientados opostos, assim obtendo um grafo orientado $G' = (V, A')$. Atribuir a cada arco $A \cap A'$ um

limite inferior 1 e a cada arco $A' \setminus A$ um limite inferior 0. Atribuir também a cada arco de A' um limite superior de 1.

Passo 2: Usando um algoritmo de fluxo em redes, determinar uma circulação viável em G. Fazer x_{ij} ser um fluxo no arco $< v_i, v_j >$.

Passo 3: Orientar alguns elos de G do seguinte modo: se $(v_i, v_j) \in E$, $x_{ij} = 1$ e $x_{ji} = 0$, orientar (v_i, v_j) de v_i a v_j.

Um procedimento para orientar completamente um grafo simétrico, pode ser colocado como segue.

Passo 1: Se todos os elos são orientados. Parar.

Passo 2: Fazer v ser o vértice com pelo menos um elo incidente não orientado (v, w). Fazer $v_1 = v$ e $v_2 := w$.

Passo 3: Orientar (v_1, v_2) de v_1 a v_2. Se $v_2 = v$, ir ao passo 1.

Passo 4: Fazer $v_1 := v_2$ e identificar um elo (v_1, v_2) incidente a v_1. Ir ao passo 3.

[Nobert & Picard 91] propuseram uma formulação matemática onde há apenas uma variável y_{ij} associada a cada elo de E. A solução de programação inteira portanto não especifica a dimensão dos elos. São impostas restrições para que o grafo aumentado satisfaça às condições necessárias e suficientes de unicursalidade, ou seja o grafo deve ser par e balanceado.

Seja um subconjunto próprio qualquer S de V, os conjuntos:

$$\begin{aligned} A^+(s) &= \{< v_i, v_j > \in A : v_i \in S, v_j \in V \setminus S\} \\ A^-(s) &= \{< v_i, v_j > \in A : v_i \in V \setminus S, v_j \in S\} \\ E(s) &= \{(v_i, v_j) \in E : v_i \in S, v_j \in V \setminus S \text{ ou } v_i \in V \setminus S, v_j \in S\} \end{aligned}$$

e seja $u(S) = |A^+(s)| - |A^-(s)| - |E(s)|$. Portanto, se $S = \{v_k\}$, então $A^+(s) = A^+(k)$, $A^-(s) = A^-(k)$, e $E(s) = E(k)$.

As constantes p_k e as variáveis z^k são definidas como seguem, e apenas uma variável y_{ij} é definida para cada $(v_i, v_j) \in E$. A variável y_{ij} representa o número de cópias do elo (v_i, v_j) que devem ser adicionadas ao grafo para torná-lo euleriano. A formulação é dada a seguir.

$$\textbf{(PCC misto-1)} \text{ Minimizar } \sum_{<v_i,v_j> \in A} c_{ij} x_{ij} + \sum_{(v_i,v_j) \in E} c_{ij} x_{ij} \qquad \text{(VI.11)}$$

sujeito a,

$$\sum_{<v_i,v_j>\in A^+(s)} x_{ij} + \sum_{(v_i,v_j)\in E} y_{ij} = 2z_k + p_k \quad (v_k \in V) \tag{VI.12}$$

$$-\sum_{<v_i,v_j>\in A^+(s)} x_{ij} + \sum_{<v_i,v_j>\in A^-(s)} x_{ij} + \sum_{(v_i,v_j)\in E} y_{ij} \geq u(S),\ S \subset V,\ s \neq \emptyset \tag{VI.13}$$

$$z_k,\ x_{ij},\ y_{ij} \in \mathcal{Z}_+ \tag{VI.14}$$

Nesta formulação, as restrições VI.13 forçam que todos os subconjuntos próprios não vazios S e V sejam balanceados, isto é feito pela imposição de que o número suficiente de arcos e elos serão introduzidos para compensar a falta de balanceamento de $u(S)$. A inclusão destas restrições foi feita como uma forma generalizada no tratamento das desigualdades de *blossons*.

$$\sum_{<v_i,v_j>\in A^+(s)} x_{ij} + \sum_{<v_i,v_j>\in A^-(s)} x_{ij} + \sum_{(v_i,v_j)\in E} y_{ij} \geq 1,\ S \subset V,\ V \text{ é ímpar}$$

[Eiselt et al 95i] mostraram que [Nobert & Picard 91] resolveram instâncias de grafos com vértices variando entre $16 \leq |V| \leq 225$ e o número de arcos variando em torno de $2 \leq |A| \leq 5569$ e elos em torno de $15 \leq |E| \leq 4455$ em tempo razoável.

Uma outra abordagem foi explorada por [Sherafat 88], a qual não considera inicialmente as condições de unicursalidade do grafo, a idéia central é buscar a unicursalidade ao longo do processo de encaminhamento da solução. [Sherafat 88] aplicou um modelo de fluxo para um grafo transformado, onde os elos são convertidos em *pseudo-ramos*, figura VI.9. A direção do fluxo nos pseudo-ramos é considerada na arborescência da solução do PCC misto, elemento que facilita o cálculo do percurso do carteiro.

No problema de fluxo a estrutura dos pseudo-ramos, figura VI.9, permite a passagem de fluxo de x_i a x_j, via os arcos $< x_j, v_r >$, $< v_r, u_r >$ e $< u_r, x_i >$ ou na direção contrária via arcos $< x_i, v_r >$, $< v_r, u_r >$ e $< u_r, x_j >$. Em ambas as direções o fluxo existirá, obrigatoriamente, passando por $< u_r, v_r >$. A este arco associa-se o custo de (x_i, x_j).

Chamando de $s \in S$ os arcos dos pseudo-ramos, o grafo transformado G' terá então novos vértices, $V' = \{X \cup V \cup U\}$, e A' como o novo conjunto de arcos. O seguinte modelo se aplica ao problema.

$$\textbf{(PCC misto-2) Minimizar} \sum_{<x_i,x_j>\in A} c_{ij} f_{ij} + \sum_{s\in S} c_{ij} f_{ij} \tag{VI.15}$$

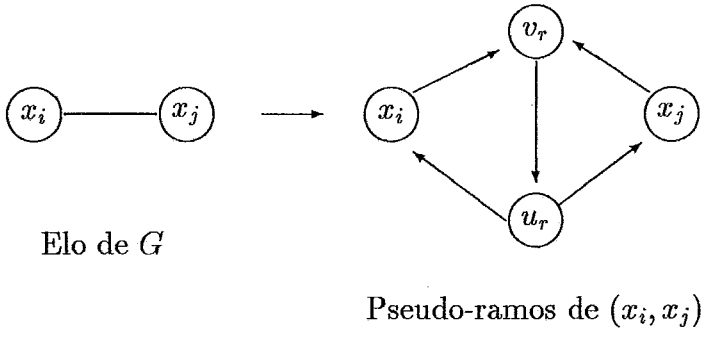


Figura VI.9: Conversão de um elo em pseudo-ramos não unicursais.

sujeito a,

$$f_{ij} \geq 1, \ \forall \ < x_i, x_j > \in \ A \cup S \tag{VI.16}$$

$$\sum_{x_i \in V'} f_{ij} \ - \sum_{x_k \in V'} f_{jk} \ = \ 0, \ \forall x_j \in V' \tag{VI.17}$$

$$f_{ij} \in \mathcal{Z}^+, \ \forall \ < x_i, x_j > \in A \cup S \tag{VI.18}$$

Utilizando um algoritmo de fluxo de custo mínimo tipo primal-dual-simplex para redes ou *Out-of-Kilter*, [Bazaraa et al 90], obtém-se uma tentativa no direcionamento dos ramos de modo a se achar as réplicas de arcos e elos visando um circuito completo de custo mínimo. Na solução ótima, haverá pelo menos uma unidade de fluxo circulando em cada arco e pseudo-ramo. Em cada pseudo-ramo l'_{ij}, o fluxo poderá passar de x_i a x_j, ou de x_j a x_i, ou eventualmente em ambos os sentidos. Entretanto, do modo como o modelo foi proposto, uma *circulação de fluxo triangular* não desejável em alguns pseudo-ramos poderá ocorrer, ver figuras VI.10 e VI.11.

Duas observações são feitas quanto às propriedades de G'. A primeira é relativa ao direcionamento da passagem de fluxo nos pseudo-ramos. Se num arco $< x_i, x_j > \in A$ exige-se uma unidade de fluxo, ela deverá achar um caminho Γ que ligue o vértice x_j ao x_i, para fechar o circuito. Γ poderá conter alguns pseudo-ramos, os quais serão forçosamente orientados conforme a direção do fluxo.

A segunda observação é quanto a passagem adicional de fluxo nos pseudo-ramos já orientados. Enquanto houver a possibilidade de direcionamento de outros ainda não orientados, uma vez que a simples orientação de tais ramos não acarretam em acréscimo no custo da solução, passar por eles uma unidade de fluxo. Este procedimento promove o direcionamento de uma maior quantidade de fluxo possível nos pseudo-ramos.

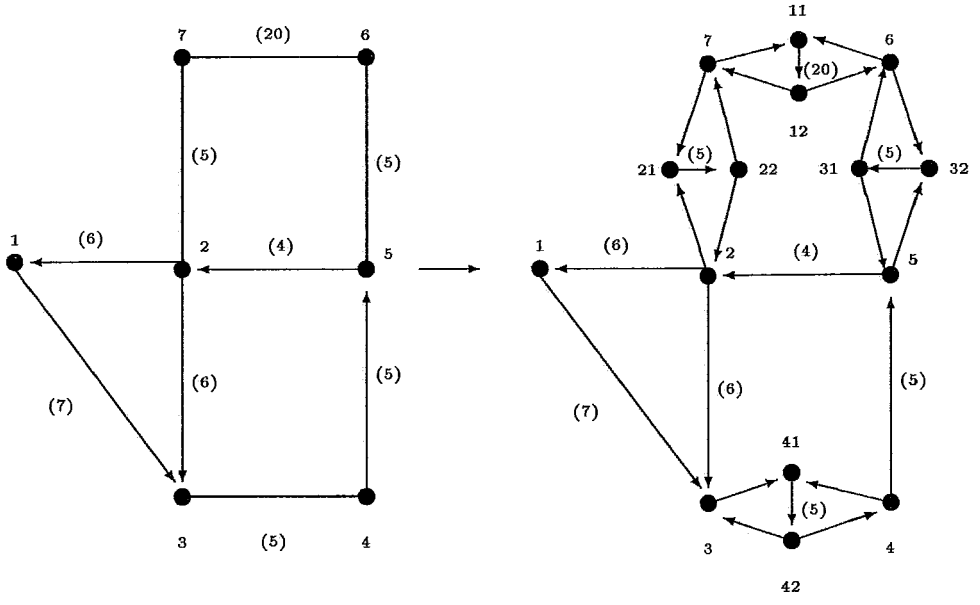


Figura VI.10: Grafo onde a aplicação direta do algoritmo de fluxo de custo mínimo atinge a solução do problema sem a formação de *circuitos triangulares*, custo da rota ótima=73.

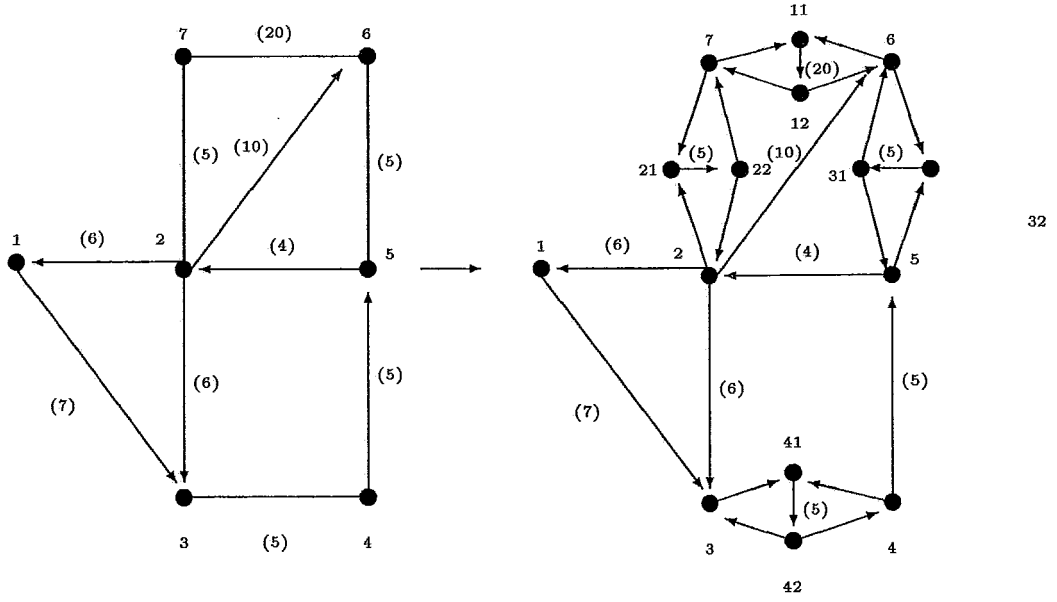


Figura VI.11: Grafo onde a aplicação direta do algoritmo de fluxo de custo mínimo não atinge a solução do problema sem a formação de *circuitos triangulares*, o fluxo triangular acontece em (6,32,31,6), ao redirecioná-lo a rota ótima é logo atingida com custo 92.

Seja o PCC misto-2, se na sua solução todos os pseudo-ramos forem orientados, a solução ótima foi obtida, ou seja: as ligações do grafo G deverão ser multiplicadas/orientadas conforme o fluxo obtido para o grafo G'. Porém, se alguns pseudo-ramos permanecerem com *fluxo triangular*, recorre-se a uma rotina de B&B.

Considerando $r \in R_o$, onde R_o representa o conjunto de todos os pseudo-ramos com fluxo triangular. Como foi visto, r corresponde a um ramo $l_{ij} \in R$ e é constituido por um conjunto de cinco arcos (dos pseudo-ramos). Fixando a direção de fluxo nesse ramo, num dos sentidos $< x_i, x_j >$ ou $< x_j, x_i >$, isto é:

$$\text{(i)} \begin{cases} f(x_i, v_r) \geq 1 \\ f(u_r, x_i) \geq 1 \end{cases} \quad \text{(ii)} \begin{cases} f(x_j, v_r) \geq 1 \\ f(u_r, x_i) \geq 1 \end{cases}$$

Inserindo as restrições (i) ou (ii) ao problema original, duas situações poderão ocorrer novamente. A primeira é quando a solução é viável para todos os pseudo-ramos, neste caso se encontrou um limite superior do problema. A segunda deve-se a um novo R_1 de pseudo-ramos com fluxo triangular que foi obtido, para este caso tem-se que repetir o procedimento, fixando a direção de fluxo para um novo $r \in R$. Tal procedimento deverá prosseguir numa árvore binária até que a solução seja ótima.

Um algoritmo exato para o PCC misto-2 considerando as colocações acima pode ser descrito como segue.

Algoritmo de Sherafat para o PCC-Misto

Passo 1: Resolver o problema PCC misto-2, $P(0)$. Se o fluxo é viável em todos os pseudo-ramos, a solução ótima foi encontrada. Ir ao passo 5. Caso contrário, fazer o limite superior de custo $LS := \infty$. Fazer $i := 0$ e ir ao passo 2.

Passo 2: Branching - Escolher um pseudo-ramo $r_k \in R_i$ para direcionar. Fazer $i := i + 1$. O novo problema a ser resolvido $P(i)$ é constituido por $P(i-1)$ acrescido de,

$$f(x_i, v_{r_k}) \geq 1$$
$$f(u_{r_k}, x_j) \geq 1$$

ir para o passo 3.

Passo 3: Bounding - Resolver o problema $P(i)$. Calcular o limite inferior do custo de circulação, LI.

- Se $LI \geq LS$ ir para o passo 4.
- Se $LI < LS$ e a solução corrente é viável, fazer $LS := LI$, armazenar o vetor de fluxo e ir para o passo 4.

- Se $LI < LS$ mas a solução não é viável, ir ao passo 2.

Passo 4: Backtracking - Seja $t \in R_j$ o último pseudo-ramo orientado, mas ainda não reorientado; j representa um estágio em que t foi orientado. Redirecionar t no sentido contrário. Fazer $i := i + 1$. O novo problema $P(i)$ é $P(j+1)$ acrescido por,

$$f(x_i, v_{r_t}) \geq 1$$

$$f(u_{r_t}, x_j) \geq 1$$

vá para o passo 3. Se não há nenhum pseudo-ramo t não reorientado, a solução ótima foi encontrada. Ir ao passo 5.

Passo 5: Gerar o grafo aumentante de G a partir da solução armazenada no último vetor de fluxo registrado. Terminar.

No algoritmo de [Sherafat 88] os pseudo-ramos que resistem ao direcionamento não é fixo, na verdade se limita ao número de problemas que serão criados no algoritmo B&B. Cada pseudo-ramo orientado nos passos 2 a 4 deverá fechar um novo ramo orientado em alguns $r \in R_i$, e/ou duplicar outros que já têm fluxo. Pelas razões já expostas, os ramos R_i por não aumentarem o custo teriam mais do que os outros, possibilidades de atraírem esse fluxo e se autodirecionarem. Desta forma, os pseudo-ramos sem fluxo viável se esgotam muito rapidamente. Além do mais, a solução do problema de fluxo $P(i)$, no passo 3 do algoritmo, geralmente requer pouco esforço computacional, uma vez que o problema $P(i-1)$ é resolvido na etapa anterior. Na maioria das vezes, poucas atualizações nas variáveis primal e dual do algoritmo primal-dual simplex são suficientes.

Nos resultados apresentados pelo autor, as soluções mostraram-se muito caras à medida que o número de elos aumenta proporcionalmente aos arcos. É de se esperar que esta técnica se torne atrativa quando se tem um grafo onde a dominância é acentuadamente maior no número de arcos.

VI.1.4 O PCC Íngreme

O PCC Íngreme é um problema da classe de problemas $\mathcal{NP}$-Árduos e contém todos os casos não-orientado, orientado e misto, como casos especiais, [Eiselt et al 95i].

Inicialmente proposto por [Minieka 79], este problema admite que alguns elos de um grafo geral $G = (V, L)$, onde $L = (E, A)$ os custos de travessia dos elos dependem do sentido que se toma. Mais formalmente, seja E o conjunto de elos de G, O conjunto formado pelos custos dos elos é dado por $C = \{(v_i, v_j) \in E : c_{ij} \neq c_{ji}\}$.

Em alguns casos o problema pode ser resolvido em tempo polinomial, isto acontece quando se tem as duas orientações de qualquer ciclo de G com o mesmo comprimento. A rota do Carteiro Íngreme determina-se tomando de G um grafo não orientado G' com os custos dos elos $(c_{ij} + c_{ji})/2$, resolver o PCC não orientado e determinar a rota de Euler.

Uma estrutura poliédrica do PCC Íngreme foi estudada por [Win 89] e por [Grötschel & Win 92], para um grafo não orientado. Seja $G = (V, L)$ o grafo e definindo por $\delta(i)$ o conjunto de elos incidentes ao vértice v_i, e $E(S)$ como o conjunto de elos (v_i, v_j) com $v_i \in S$, $v_j \notin S$, onde $S \subset V$. Seja também x_{ij} uma variável inteira que indica quantas vezes o elo (v_i, v_j) é atravessado de v_i a v_j na solução ótima do PCC Íngreme. O problema pode ser formulado como segue.

$$\text{(PCC Íngreme-1) Minimizar} \quad \sum_{(v_i,v_j)\in L} (c_{ij}x_{ij} + c_{ji}x_{ji}) \tag{VI.19}$$

sujeito a,

$$x_{ij} + x_{ji} \geq 1 \quad (v_i, v_j) \in L \tag{VI.20}$$

$$\sum_{(v_i,v_j)\in\delta(i)} (x_{ij} - x_{ji}) = 0 \quad (v_i \in V) \tag{VI.21}$$

$$x_{ij}, x_{ji} \geq 0 \quad (v_i, v_j) \in L \tag{VI.22}$$

$$x_{ij}, x_{ji} \in \mathcal{Z}_+ \quad (v_i, v_j) \in L \tag{VI.23}$$

Seja P o poliedro formado pelos vetores $x = (x_{ij}, x_{ji})$ satisfazendo as equações de VI.20, VI.21 e VI.22. Então G é euleriano se e somente se todo vértice de P é inteiro; além disto, os componentes dos vértices de P são sempre 0, 1/2, ou um inteiro positivo. Também, se G é euleriano, então VI.20-VI.22 dá uma descrição completa da envoltória convexa dos vetores incidentes nas rotas de G, chamadas de **PI**(G), [Grötschel & Win 88]. [Gendreau et al 90] mostraram que um resultado proposto por [Minieka 79] para o PCC misto aplica-se ao PCC Íngreme: *em qualquer solução ótima do PCC Íngreme, um dos três casos seguintes ocorre para a travessia de qualquer elo* (v_i, v_j):

1. $x_{ij} = 0$ e $x_{ji} \geq 1$,
2. $x_{ij} \geq 1$ e $x_{ji} = 0$, ou
3. $x_{ij} = x_{ji} \geq 1$.

O que sugere um esquema de ramificação nos métodos *B&B* de três modos.
[Grötschel & Win 88] mostraram que cada uma das desigualdades VI.20 e VI.22 definem as facetas de **PI**(G) se e somente se (v_i, v_j) não é uma ponte de G. Além disso para qualquer $|E(S)|$ ímpar, as seguintes *desigualdades de corte ímpar* são

válidas com respeito ao $\mathbf{PI}(G)$, e definem as faces de $\mathbf{PI}(G)$ se e somente se os subgrafos G induzidos por S e $V \setminus S$ são conexos:

$$\sum_{(v_i,v_j)\in E(S)} (x_{ij} + x_{ji}) \geq |E(S)| + 1, \ (S \subset V) \tag{VI.24}$$

$$\sum_{v_i\in S,\ v_j\notin S} x_{ij} \geq \frac{1}{2}(|E(S)| + 1) \ (S \subset V) \tag{VI.25}$$

e

$$\sum_{v_i\in S, v_j\notin S} x_{ji} \geq \frac{1}{2}(|E(S)| + 1) \ (S \subset V) \tag{VI.26}$$

Desigualdades adicionais válidas também são consideradas em [Grötschel & Win 88]. [Grötschel & Win 92] observaram que apesar do Problema de Programação Linear definido por VI.20-VI.22 e VI.24-VI.26 conter um número de restrições exponencial, ele pode ser resolvido em tempo polinomial no tamanho da entrada de um dado PCC Íngreme fazendo uso do procedimento de [Padberg & Rao 82] que satisfaz às restrições de *corte-ímpar*.

[Grötschel & Win 92] dividiram um algoritmo de planos de corte para resolver o Problema Linear, e o aplicaram na resolução de instâncias do PCC Íngreme em grafos com $52 \leq |V| \leq 264$ vértices, tendo entre $78 \leq |E| \leq 489$ elos. A solução do Problema Linear atingiu a solução ótima em 31 dos 36 problemas. Quando o Problema Linear falha numa solução inteira, rotas viáveis do PCC Íngreme podem ser obtidas arredondando as variáveis ao seu maior inteiro, algumas destas variáveis assumem porém valor zero.

Uma formulação alternativa pode ser derivada da formulação proposta no PCC misto-2. Admitindo que os pseudo-ramos são alterados da sua forma original da figura VI.9 para a situação mostrada na figura VI.12. Fazendo o custo no arco $< v_1, u >$ ser c_{ij} e o custo no arco $< v_2, u >$ ser c_{ji}, e mantendo-se inalteradas as condições do modelo. Tem-se daí uma nova aproximação para o PCC Íngreme.

$$\text{(PCC Íngreme-2) Minimizar} \sum_{<x_i,x_j>\in A} c_{ij} f_{ij} + \sum_{<s_{v_1},s_u>\in S} c_{s_{v_1}s_u} f_{s_{v_1}s_u} + \sum_{<s_{v_2},s_u>\in S} c_{s_{v_2}s_u} f_{s_{v_2}s_u} \tag{VI.27}$$

sujeito a,

$$f_{ij} \geq 1, \ \forall \ < x_i, x_j > \in \ A \tag{VI.28}$$

$$f_{s_{v_1}s_u} + f_{s_{v_2}s_u} \geq 1, \ \forall \ < s_{v_1}, s_u >, < s_{v_2}, s_u > \in S \tag{VI.29}$$

$$\sum_{x_i\in V'} f_{ij} - \sum_{x_k\in V'} f_{jk} = 0, \ \forall x_j \in V' \tag{VI.30}$$

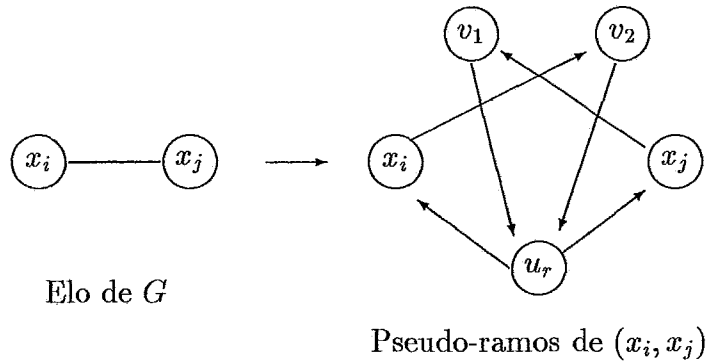


Figura VI.12: Conversão de um elo em pseudo-ramos não unicursais para o PCC Íngreme.

$$f_{ij} \in \mathcal{Z}^+, \ \forall \ < x_i, x_j > \in \ A \cup S \tag{VI.31}$$

As restrições VI.29 determinam uma condição de fluxo pelo menos unitário através dos arcos centrais dos pseudo-ramos. O modelo possui um número de variáveis que supera a formulação do PCC misto-2 do número de elos de G, porém se adapta perfeitamente ao algoritmo do carteiro chinês misto proposto por [Sherafat 88].

VI.1.5 O PCC Hierárquico

Este problema, também da classe de problemas $\mathcal{NP}$-Árduos é definido sobre um grafo geral $G = (V, L)$, onde V contém uma fonte s e um sumidouro t. As ligações de L são incidentes a s ou t, e são particionadas de tal modo que $\{l_1, \ldots, l_k\}$ devem ser ordenados em uma relação de ordem $\prec$ sobre alguns dos elementos da partição. O PCC Hierárquico consiste em determinar uma travessia de menor custo de G iniciando em s e terminando em t, servindo os arcos da partição de tal modo que se $l_p \prec l_q$[5], então todos as ligações l_p são servidos antes das ligações l_q, porém ligações de l_q podem ser atravessadas antes de l_p. A precedência está pois no serviço, não na travessia. Como foi visto, este problema surge na remoção de neve, na varrição de rua, dentre outros, [Dror et al 87].

Considerando os subgrafos $G_p = (V_p, L_p)$ induzidos pelos conjuntos L_p. Se cada subgrafo é f-conexo e a relação de ordem é completa, ou seja, uma ordem total $L_1 \prec L_2 \prec \ldots \prec L_k$ é especificada, o problema pode ser resolvido em $O(k|V|^5)$. O algoritmo que resolve este problema é descrito como segue.

[5] l_p precede l_q

Construir um grafo orientado e acíclico $\overline{G} = (\overline{V}, \overline{A})$ como segue. Definir $\overline{v_i}$ como um conjunto contendo uma cópia de cada vértice incidente a um arco de A_i, e seja $\overline{V} = \{s, t\} \cup \overline{V_1} \cup \ldots \cup \overline{V_k}$. O conjunto do arco $\overline{A}$ é composto por todos os arcos formados pelo vértice s atingindo os vértices de $\overline{V_1}$ e dos vértices de $\overline{V_k}$ atingindo t, e de todos os arcos formado pelos vértices de $\overline{V_p}$ para os vérticas de $\overline{V_{p+1}}$ $(p = 1, \ldots, k-1)$. O custo c_{uv} de um arco $(u, v) \in \overline{A}$ pode ser determinado como segue: se $u = v$, $u \in \overline{V_p}$ e $u \in \overline{V_{p+1}}$ para $(p = 1, \ldots, k)$ e $\overline{V_{k+1}} = \{t\}$, então c_{uv} é o valor do menor caminho euleriano ou da cadeia mínima iniciando em u e finalizando em v, cobrindo todos os arcos de s a t em $\overline{G}$.

A complexidade deste procedimento é determinada pelo cálculo dos custos dos arcos em $\overline{G}$. Uma vez que qualquer vértice de $\overline{V}$ pode pertencer a um limite de k conjuntos de vértices V_p, há $O(k|V|^2)$ destes arcos. Existem algoritmos que calculam o custo de cada arco em $O(|V|^3)$, se G é não orientado ou orientado.

Quando os subgrafos G_p não estão conexos ou se a ordem de relação é apenas parcial, o problema se torna $\mathcal{NP}$-Árduo, [Eiselt et al 95i]. [Gelinas 92] descreve um algoritmo enumerativo exato para o PCC Hierárquico com vértices especificados iniciais e finais, no qual a ordem de relação é geral, e todos os subgrafos G_p são fortemente conexos. O problema é resolvido através de um algoritmo de programação dinâmica no qual os estados correspondem a subgrafos já atravessados, e o último vértice do caminho atual. Nas operações de remoção de neve em Montreal, instâncias reais com $|V| = 171$ e $|A| = 544$ e $k = 4$ foram resolvidas na otimalidade.

VI.2 O Problema do Carteiro Rural

No PCR o objetivo é a construção de uma rota que passa por um conjunto de ligações desejadas $R \subseteq L$. A complexidade deste problema foi definida por [Lenstra & Rinooy Kan 81] como sendo $\mathcal{NP}$-Árduo, apesar do PCC nos casos não orientado e orientado possuirem solução em tempo polinomial, para $R = L$.

A versão mista do PCR onde L é um conjunto de ligações e R é o conjunto exclusivamente formado por arcos, é chamada de Problema de *Stacker Crane* - PSC, sendo também da classe de problemas $\mathcal{NP}$-Árduo, [Frederickson et al 78].

Se além de se considerar um R de G tem-se associado a R valores de demanda $0 \leq q_{ij} \leq Q$ nas ligações $v_i v_j$, e há uma frota de veículos homogêneos de capacidade limitada Q por veículo, que parte de um ponto inicial (depósito) de G e a ele retorna a um custo total mínimo de percurso, tem-se o Problema de Percurso (Roteamento) em Arcos com Demanda - PRAD, [Golden & Wong 81]. O PRAD também é $\mathcal{NP}$-Árduo, [Lenstra & Rinooy Kan 81], pois trata-se de um caso especial do PCR, uma vez que se considera $Q = |R|$, $q_{ij} = 1$ se $v_i, v_j \in R$ e $q_{ij} = 0$ se

$v_i, v_j \notin R$. O PRAD também pode ser visto como uma generalização do Problema do Carteiro Chinês Capacitado onde a diferença básica é que $q_{ij} > 0$, $\forall v_i, v_j \in L$, ou seja, $R = L$ e toda ligação possui demanda, [Christofides 73]. Como em geral as aplicações de roteamento em arcos têm restrições de capacidade, o PRAD é tido como dos mais importantes da área.

Como o PCR é $\mathcal{NP}$-Árduo, o uso de heurísticas para a geração do grafo aumentante é bastante comum. Em geral, as heurísticas consideram algoritmos de *Matching* ou de *Árvores Geradoras Mínimas* com o objetivo de gerar um grafo aumentante que satisfaça às restrições de unicursalidade.

Nos algoritmos exatos do PCR, a geração de grafos eulerianos exploram duas das técnicas comumente usadas no PCV. Quando os grafos são orientados, o problema é formulado como um PPL contendo um conjunto muito grande de restrições. Este problema é resolvido usando *Branch & Cut*, ou seja, uma relaxação contínua do problema é inicialmente resolvida e cortes válidos são gerados quando se encontram violações. A ramificação é usada para ganhar novamente a integralidade e os cortes devem ser novamente adicionados aos subproblemas. Este método é muito usado nas soluções exatas do PCV, ver capítulo III.

Para resolver problemas orientados, alguns autores usam a resolução de uma relaxação da arborescência expandida mínima, ligada ao algoritmo *B&B*, seguindo as idéias gerais de solução exata do PCV.

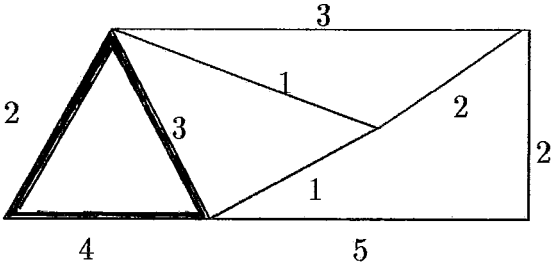
VI.2.1 O PCR Não Orientado

No PCR não orientado $L = E$. O caso mais simples ocorre quando $\overline{G} = (V, R)$ é conexo. Neste caso, o problema pode ser resolvido através do cálculo das cadeias mais curtas (em G) entre os vértices de grau ímpar (algumas destas cadeias devem ter elos em $L \setminus R$), e em seguida proceder com o algoritmo do PCC não orientado, [Orloff 74].

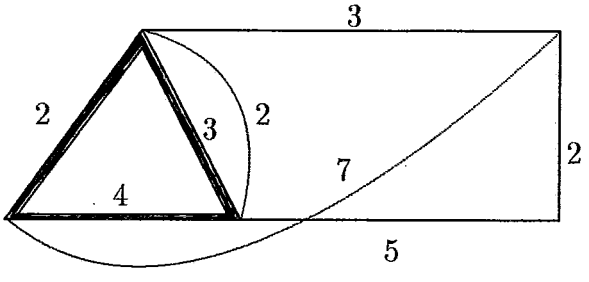
Em geral o PCR é resolvido sobre um grafo modificado $G' = (V', L')$ definido como segue: $V' = \{v_i \in V \ : \ (v_i, v_j) \in R, \text{ para algum } v_j \in V\}$ e L' é obtido inicialmente pela inclusão em L de um elo (v_i, v_j), então retira-se todos os elos $(v_i, v_j) \in L \setminus R$ que satisfazem à desigualdade triangular $c_{ij} = c_{ik} + c_{kj}$ para algum k, assim como um dos elos paralelos se tiverem o mesmo custo.

A idéia geral da transformação de G em G' pode ser vista na figura VI.13. No grafo G' o conjunto R induz p componentes conexas $G_1, \ldots, G_p$ com um conjunto de vértices respectivos $V_1, \ldots, V_p$ formando uma partição de V'. Caso a matriz C satisfaça a desigualdade triangular, uma heurística aproximativa ($\epsilon = 3/2$) proposta por [Frederickson 79], baseado nas idéias do algoritmo de [Christofides 76]

para o PCV, pode ser usada. A seguir tem-se a sua descrição.



(a) Grafo original G



Grafo transformado G'

Figura VI.13: Grafo modificado para o PCR com G' conexo.

Algoritmo Heurístico para o PCR não Orientado

Passo 1: (Árvore Geradora Mínima) Construir a árvore geradora mínima T contendo $G_1, \ldots, G_p$ e fazer $l(T)$ seu comprimento ótimo. Chamar de $l(R)$ a soma dos comprimentos de todos os elos de R e por z^* uma solução ótima do PCR. Então $l(T) + l(R) \leq z^*$.

Passo 2: (1-Matching) Determinar o *1-Matching* de custo mínimo entre todos os vértices de grau ímpar do grafo induzido por $R \cup T$. Seja M o conjunto de elos incluídos no *1-matching* ótimo, e seja $l(T)$ o seu comprimento total.

Passo 3: (Ciclo Euleriano) Uma solução do PCR é dada pelo ciclo euleriano sobre o grafo induzido por $R \cup T \cup M$. Se C satisfaz à desigualdade triangular, pode-se mostrar que $l(M) \leq z^*/2$, pelos mesmos argumentos de [Christofides 76]. Portanto o comprimento do ciclo euleriano não ultrapassa a $3z^*/2$.

Apesar do expressivo resultado da heurística de [Frederickson 79], sua fundamental importância está nos elementos que deu para as formulações matemáticas que foram sugeridas para o PCR.

Duas formulações de PLI foram propostas para o PCR não orientado. Na primeira, sugerida por [Christofides et al 81], as varáveis x_{ij} são definidas como segue: Se $(v_i, v_j) \in R$, x_{ij} é o número de vezes que o elo (v_i, v_j) será replicado na solução ótima do PCR, ou seja, o elo (v_i, v_j) será percorrido $1 + x_{ij}$ vezes. Se $(v_i, v_j) \in L' \setminus R$, x_{ij} é o número de vezes que o elo (v_i, v_j) é percorrido. A formulação pode ser colocada como segue.

$$\textbf{(PCR} \text{ Não Orientado-1) Min} \sum_{(v_i,v_j)\in R} c_{ij}(1 + x_{ij}) + \sum_{(v_i,v_j)\in L'\setminus R} c_{ij}x_{ij} \qquad \text{(VI.32)}$$

sujeito a,

$$\sum_{\substack{(v_i,v_j) \in R \\ j > i}} (1 + x_{ij}) + \sum_{\substack{(v_i,v_j) \in R \\ j < i}} (1 + x_{ji}) + \sum_{\substack{(v_i,v_j) \in L' \setminus R \\ j > i}} x_{ij} + \sum_{\substack{(v_i,v_j) \in L' \setminus R \\ j < i}} x_{ji} \equiv 0 \pmod 2 \ (v_i \in V') \qquad \text{(VI.33)}$$

$$\sum_{\substack{v_i \in S \\ v_j \notin \overline{S}}} x_{ij} \geq 0 \ (S = \bigcup_{k\in P} V_k, \ \overline{S} = (\bigcup_{k=1}^{p} V_k) \setminus S, \ P \subset \{1, \ldots, p\}) \qquad \text{(VI.34)}$$

$$x_{ij} \in \mathcal{Z}_+ \ (v_i, v_j) \in L' \qquad \text{(VI.35)}$$

As restrições VI.33 garantem que o grau de todo vértice da solução é par, enquanto as restrições VI.34 garantem que um ciclo ótimo de todos os componentes conexos serão ligados. Isto é feito de forma relativamente fácil quando se limita superiormente de 1 se $(v_i, v_j) \in R$, e por 2 se $(v_i, v_j) \in L' \setminus R$. As restrições VI.33 podem ser trocadas por novas restrições VI.36, fazendo-se o lado direito igual a $2z_i$, com

$$z_i \in \mathcal{Z}_+ \ (v_i \in V') \qquad \text{(VI.36)}$$

Como no algoritmo de [Held & Karp 71], [Christofides et al 81] fazem uma derivação do limite superior do problema pela incorporação de restrições do tipo VI.36 no modo da relaxação lagrangeana via função objetivo, e por observarem que o subproblema definido pelas variáveis x_{ij} associadas a $L' \setminus R$ é um problema de AGM sobre um grafo cujos vértices correspondem aos componentes conexos de

G'. Também, usando a AGM, eles derivam um limite superior para o PCR emparelhando vértices de grau ímpar. Estes procedimentos foram incorporados a um procedimento *B&B* que foi testado para instâncias geradas randomicamente cujas características variavam conforme os vértices, ligações, elos escolhidos e componentes conexas, $9 \leq |V| \leq 184$, $13 \leq |L| \leq 184$, $4 \leq |R| \leq 78$ e $2 \leq p \leq 8$.

Uma segunda formulação foi proposta por [Corberán & Sanchis 91]. Seja $E_i = \{(v_i, v_j) \in L'$ o conjunto de todos os elos incidentes a v_i em G'. Um vértice é chamado R-par (respectivamente R-ímpar) se ele é incidente a um número par (ímpar) de elos em R. As variáveis x_{ij} são definidas como no modelo não orientado-1. O problema pode ser formulado como segue.

$$\textbf{(PCR} \text{ Não Orientado-2) Min} \sum_{(v_i,v_j)\in R} c_{ij}(1 + x_{ij}) + \sum_{(v_i,v_j)\in L'\setminus R} c_{ij}x_{ij} \qquad \text{(VI.37)}$$

sujeito a,

$$\sum_{(v_i,v_j)\in L_i} x_{ij} = 0 \pmod 2 \quad (v_i \in V',\ v_i \text{ é } R\text{-par}) \qquad \text{(VI.38)}$$

$$\sum_{(v_i,v_j)\in L_i} x_{ij} = 1 \pmod 1 \quad (v_i \in V',\ v_i \text{ é } R\text{-ímpar}) \qquad \text{(VI.39)}$$

$$\sum_{\substack{v_i \in S, v_j \notin \overline{S} \\ \text{ou} \\ v_i \in S, v_j \notin \overline{S}}} x_{ij} \geq 2 \ (S = \bigcup_{k\in P} V_k,\ \overline{S} = (\bigcup_{k=1}^{p} V_k) \setminus S,\ P \subset \{1,\ldots,p\}) \qquad \text{(VI.40)}$$

$$x_{ij} \in \mathcal{Z}_+ \quad (v_i, v_j) \in L' \qquad \text{(VI.41)}$$

O mérito desta formulação está em que os autores identificaram uma família de faces do politopo de uma envoltória convexa de soluções definidas pelas equações de VI.38-VI.40. Os autores também mostraram que todas as desigualdades de indução de faces para o chamado PCV-(de rua) também são faces para o PCR não orientado. Segundo o trabalho de [Padberg & Grötschel 85], [Corberán & Sanchis 91] incorporaram rotinas de geração de faces através de um procedimento *B&B*, aplicando às 24 instâncias do trabalho de [Christofides et al 81], onde 23 deles obtiveram a solução ótima no nó raiz da árvore de busca, usando apenas os planos de corte. Dois problemas novos, extraídos de uma aplicação real na cidade de Albaida (Valência) também foram resolvidos sem ramificação.

VI.2.2 O PCR Orientado

O PCR Orientado considera que $L = A$. O problema se reduz ao PCC orientado sempre que $\overline{G} = (V, R)$, sendo $R = A$ e G conexo, como de forma semelhante acontece no caso não orientado. Já para o caso mais geral, o problema é resolvido sobre um grafo modificado $G' = (V', L')$ construído como no caso não orientado, exceto que os arcos $< v_i, v_j >$ adicionados têm um sentido de travessia definido e têm um custo c_{ij} igual ao caminho mais curto de v_i a v_j. Os componentes conexos $G_1, \ldots, G_p$ com um conjunto respectivo de vértices são definidos como antes. Para o PCR orientado, a seguinte heurística foi proposta, [Christofides et al 86].

Algoritmo Heurístico para o PCR Orientado

Passo 1: (Arborescência Expandida Mínima) Construir a menor arborescência expandida enraizada em um vértice arbitrário, e conectando $G_1, \ldots, G_p$. Tomar $\overline{G}$ o grafo resultante.

Passo 2: (Problema de Transporte) Assim como no caso orientado do PCC, gerar um grafo euleriano de $\overline{G}$ pela adição de arcos de menor custo tal que o número de arcos entrando em cada vértice torne-se igual ao número de arcos saindo.

Passo 3: (Circuito Euleriano) Determinar um circuito euleriano sobre o grafo aumentado.

O mesmo procedimento pode ser repetido tomando-se cada vértice de G como raiz da AGM, selecionando-se então a solução de menor custo. Usando este procedimento os autores resolveram 23 instâncias cujo custo esteve em média 1,3 % distante da solução ótima. O maior desvio registrado foi de 5% além do ótimo, porém dez delas atingiram o ótimo.

VI.2.3 O Problema *Stacker Crane*

O problema de *Stacker Crane*, é definido sobre um grafo misto $G = (V, L)$, $L = E \cup A$, onde apenas o conjunto dos arcos deverá ser visitado, $R = A$, figura VI.14. O objetivo é a determinação de um percurso de custo mínimo que inclui cada arco R pelo menos uma vez. Neste problema o conjunto R é visto como movimentos de um guindaste (*Crane*) os quais devem ser feitos exatamente uma vez, numa direção e sentido especificados. R também pode ser visto como entregas a serem feitas por um veículo, onde o PSC busca minimizar a soma dos custos entre entregas. O PSC é equivalente ao PCV quando se considera que o custo c_{ij} de todo arco $(v_i, v_j) \in R$ é zero, em virtude desta transformação o PSC também é do tipo $\mathcal{NP}$-Árduo.

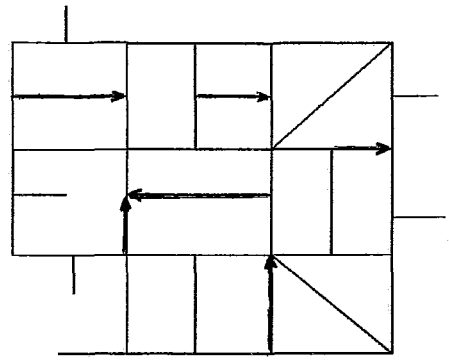


Figura VI.14: Grafo relativo a um PSC.

[Frederickson et al 78], propuseram duas heurísticas que requerem de G a satisfação das seguintes propriedades:

1. Cada vértice é incidente a pelo menos um arco de R;
2. Os c_{ij} sobre os elos satisfazem à desigualdade triangular.

Se G não satisfizer estas duas propriedades, ele pode ser transformado em um grafo G' que as satisfaça. O problema é então resolvido sobre G' e em seguida a solução pode ser interpretada sobre G, como eventualmente se faz nestes problemas. As heurísticas são chamadas de LARGEARCS e SMALLARCS. A primeira possui um desempenho melhor quando o custo dos arcos é grande comparado ao custo da solução ótima, já o desempenho é melhor na segunda quando o contrário acontece.

Heurística LARGEARCS

Passo 1: (Grafo Bipartido) Retirar de G o conjunto de arcos R orientados. Construir um grafo bipartido G' onde o primeiro conjunto de vértices corresponde aos vértices iniciais (v_i) de R e o segundo corresponde aos vértices finais (v_j) de R. Criar elos entre todos os vértices iniciais e finais que não correspondem ao mesmo arco, determinar o seu custo a partir do menor caminho entre eles.

Passo 2: (Matching) Resolver o problema de *1-matching* mínimo bipartido. Construir um grafo orientado G'' tendo um conjunto de arcos consistindo de R e todos os elos na solução do *1-matching*, orientados a partir dos vértices iniciais v_i aos finais v_j. (Isto faz com que apareçam $m \geq 1$ ciclos disjuntos compostos de elos e arcos.)

Passo 3: (AGM) Seja os m ciclos disjuntos chamados de G''_i, onde cada G''_i é representado por um único vértice n_i. Das distâncias entre os vértices do grafo original G,

$$d_{n_i,n_j} = min\{c'(x,y) \ : \ x \in R_i, \ y \in R_j\}$$

Associar a (n_i, n_j) um elo particular (x, y) que determina o custo mínimo. Encontrar a árvore expandida mínima para os nós $\{n_i \ : \ 1 \leq i \leq m\}$, usando a matriz de distância $D = (d_{n_i,n_j})$.

Passo 4: (Rota Euleriana) Construir um grafo orientado $\overline{G}$ tendo um conjunto de arcos consistindo de todos os arcos de G'', e dois arcos para cada elo da árvore expandida gerada no passo 3. Este grafo é euleriano e a solução do PSC é dada pela resolução do PCC orientado sobre $\overline{G}$.

[Frederickson et al 78] mostram que o algoritmo LARGEARCS possui complexidade $O(max\{|V|^3, |A|^3\})$. Além disso, como o custo da matriz satisfaz a desigualdade triangular, o procedimento produz uma rota de comprimento $l(T)$ satisfazendo a

$$l(T) \leq \ 3z^* - 2l(A),$$

onde z^* é o valor da solução ótima do PSC, e $l(A)$ é a soma dos comprimentos de todos os arcos de A.

No algoritmo SMALLARCS considera-se inicialmente que a solução poderá ser produzida, uma vez que G seja um grafo misto, par, simétrico e balanceado. É importante também que as propriedades sejam satisfeitas.

Heurística SMALLARCS

Passo 1: (Grafo Contraido) Representar cada arco $< t_i, h_j > \in R$ por um vértice n_i. Para cada par de vértices n_i e n_j definir $c'(n_i, n_j) = min\{c_{t_i,t_j}, c_{t_i,h_j}, c_{t_i,h_j}, c_{h_i,t_i}, c_{h_i,h_j}\}$. Executar o algoritmo de caminho mínimo entre todos os pares de vértices usando c' para encontrar a distância entre os vértices n_i e n_j. Associar a cada elo (n_i, n_j) os elos do caminho mínimo entre n_i e n_j. Tem-se então um grafo $G^\star$

Passo 2: (AGM) Encontrar a Árvore Geradora mínima sobre $G^\star$.

Passo 3: (1-Matching) Aplicar o *1-matching* de custo mínimo dos vértices de grau ímpar da árvore mínima.

Passo 4: (Reexpandindo o Grafo) Trocar todos os vértices de A. E trocar cada elo expandido e emparelhado por uma cadeia mínima correspondente ao passo 1,

obtendo pois um grafo misto $\overline{G}$. Nota-se que (v_i, v_j) é um arco de G, então ambos v_i e v_j têm o mesmo grau de paridade. Se v_i e v_j têm grau ímpar, adicionar o elo (v_i, v_j), e direcioná-lo temporariamente de v_j a v_i. Chamar G^* o grafo resultante.

Passo 5: (Primeira Rota Euleriana) Encontrar uma rota euleriana em $\overline{G}$, ingnorando as direções dos arcos pares. Se o comprimento dos arcos pares que são atravessados para trás ultrapassam $l(|\overline{A}|)/2$, reverter a direção da rota.

Passo 6: (Adicionar Arcos) Associar a cada elo a direção da rota. Se o arco (v_i, v_j) é atravessado na direção errada, adicionar dois arcos (v_j, v_i). Chamar de $\overline{\overline{G}}$ o grafo orientado resultante.

Passo 7: (Rota Euleriana Final) Determinar o circuito euleriano sobre $\overline{\overline{G}}$.

Os autores também mostraram que a complexidade do algoritmo SMALLARCS é igual à do algoritmo LARGEARCS. Como a matriz de custos satisfaz a desigualdade triangular, o comprimento de $l_S(T) \leq 1/2\ (3z^* + l(A))$.

Uma aplicação composta simultaneamente dos algoritmos LARGEARCS e SMALLARCS pode ter melhor sucesso. Ou seja, a aplicação dos dois algoritmos em um mesmo grafo com a posterior escolha da melhor solução. Chamando de $l(T) = min\{l_L(T) + l_S(T)\}$, foi mostrado que

$$l(T) = 9z^*/5.$$

Uma nova versão do PSC foi explorada por [Frederickson et al 78], considerando a situação de mais de um veículo, ou o k-PSC. O objetivo deste problema é minimizar o tamanho da maior rota. A idéia geral do algoritmo é colocada como segue.

Algoritmo Heurístico k-CRANES

Passo 1: Encontrar a solução para o 1-PSC. Tomar v_1 como o vértice inicial.

Passo 2: Para cada j, $1 \leq j \leq k$, encontrar o último vértice $v_{p'(j)}$ tal que $c(R_{v'_p(i)}) \leq L_j$.

Passo 3: Seja $r_j = L_j - c(R_{v'_p(j)})$. Para cada j, $1 \leq j \leq k$, se $(v_{p'(j)}, v_{p'(j)+1})$ e o vértice inicial de R_{j+1} é $v_{p'(j)+1}$. Suponha que $(v'_p(j), v_{p'(j)+1})$ é um arco. Se

$$c(v_1, v_{p'(j)}) + r_j \ \leq \ c(v_{p'(j)}, v_{p'(j)+1}) \ - \ r_j \ + \ c(v_{p'(j)+1}, v_1)$$

então $v_p(j) = v_{p'(j)}$ e o vértice terminal de R_j é $v_{p(j)}$. Também se $(v_{p(j)-1}, v_{p(j)})$ é um arco, então o vértice inicial de R_j é $v_{p(j)}$, e se ele é um elo então o vértice terminal de R_j é $v_{p(j)-1}$. Se

$$c(v_1, v_{p'(j)}) + r_j \;>\; c(v_{p'(j)}, v_{p'(j)+1}) \;-\; r_j \;+\; c(v_{p'(j)+1}, v_1)$$

então $v_p(j) = v_{p'(j)}$ e o vértice terminal de R_{j+1} é $v_{p(j)}$, caso contrário o vértice inicial é $v_{p(j)+1}$.

Passo 4: Para cada j, $1 \leq j \leq k$, construir uma j-ésima subrota R_j pela conecção de v_1 ao vértice inicial de R_j e o vértice terminal de R_j a v_1 com elos orientados.

[Frederickson et al 78] mostram que o algoritmo k-Cranes é polinomial de ordem igual aos algoritmos LARGEARCS e SMALLARCS, cuja solução é aproximativa em

$$\epsilon = 14/5 - 1/k.$$

VI.3 O Problema de Roteamento em Arcos com Demanda (PRAD)

No PRAD cada ligação $(v_i, v_j) \in L$ tem uma demanda não negativa ou peso q_{ij}. Admite-se que uma frota de m veículos homogêneos de capacidade Q estão baseados em um depósito localizado no vértice v_1. O PRAD consiste em determinar o percurso de custo mínimo que atende a todas ligações solicitadas com um conjunto de veículos, tal que a demanda total das ligações servidas por qualquer veículo não excede a Q. Nota-se que um veículo pode atravessar um arco sem servi-lo, mas cada arco deve ser atendido por apenas um veículo. O PRAD é uma generalização do PCC capacitado, como já foi colocado, onde $q_{ij} > 0$ para todo $(v_i, v_j) \in L$, [Christofides 73].

[Golden & Wong 81] mostraram que encontrar um algoritmo 1/2-aproximativo para um PRAD onde C é uma matriz de custos que satisfaz à desigualdade triangular, é $\mathcal{NP}$-Completo. Para mostrar este fato eles reduziram o PRAD ao Problema da Partição, provado por [Karp 72] ser $\mathcal{NP}$-Completo. [Pearn 84] e [Win 87] mostraram que este resultado se verifica quando G é um caminho. Já [Sahni & Gonzalez 76] mostraram que quando a matriz de custos não satisfaz à desigualdade triangular, a solução do PCV ϵ-aproximativo é $\mathcal{NP}$-Completo para um valor de ϵ finito e portanto isto se verifica para o PRV capacitado e para o PRAD.

Os casos que podem ser resolvidos exatamente foram pesquisados por [Assad et al 1987] e por [Busch 91]. Se G é um grafo linear, todos os veículos têm a mesma capacidade Q, e todas as ligações têm a mesma demanda $q_{ij} = 1$, então o

PCC capacitado pode ser resolvido exatamente em $O(|V|)$. Quando a frota é heterogênea, há veículos de capacidades distintas, então o problema pode ser resolvido através de um algoritmo pseudo-polinomial, [Busch 91]. Entretanto em PRADs definidos sobre grafos lineares, mas com demandas distintas, estes são $\mathcal{NP}$-Árduos, como também são aqueles cujos grafos são árvores. Por fim, [Assad et al 1987] mostraram que em PCCs capacitados com demandas idênticas que estão definidos sobre um grafo cíclico são resolvidos em $O(|V|)$; já para um grafo completo pode ser resolvido em tempo polinomial se todos os q_{ij} não são maiores que $Q/|V|$ quando $|V|$ é ímpar, e nunca maiores que $Q/(|V|-1)$ quando $|V|$ é par.

VI.3.1 Formulações do PRAD

Duas formulações de PLI foram propostas para o problema não orientado, onde $L = E$, do PRAD. [Golden & Wong 81] propuseram a primeira delas usando variáveis direcionadas, e a segunda foi proposta por [Belenguer & Benavent 91] usando variáveis não direcionadas.

Considerando as variáveis da formulação direcionada do PRAD, tem-se:

$$x_{ij}^k = \begin{cases} 1, & \text{se } (v_i, v_j) \text{ é atravessado de } v_i \text{ a } v_j \text{ pelo veículo } k \\ 0, & \text{Caso Contrário} \end{cases}$$

$$l_{ij}^k = \begin{cases} 1, & \text{se } (v_i, v_j) \text{ é servido de } v_i \text{ a } v_j \text{ pelo veículo } k \\ 0, & \text{Caso Contrário} \end{cases}$$

x_{ij}^k tem limite superior em 1 enquanto não for necessário atravessar o elo nesta direção mais de uma vez. Seja os conjuntos $P(i)$ formado pelos predecessores de v_i e $F(i)$ pelos sucessores de v_i, para todo $v_i \in V$. Se qualquer ligação (v_i, v_j) com $q_{ij} > 0$ deve ser atendida e que as demais com $q_{ij} = 0$ podem ou não ser atravessadas por algum veículo. Tomando-se m como o número de veículos, e A o conjunto dos arcos orientados a partir de E, a formulação direcionada é a que segue.

$$\textbf{(PRAD} \text{ Orientado) Minimizar } \sum_{k=1}^{m} \sum_{(v_i,v_j)\in A} c_{ij} x_{ij}^k \qquad \text{(VI.42)}$$

sujeito a,

$$\sum_{j\in P(i)} x_{ji}^k - \sum_{j\in F(i)} x_{ij}^k = 0, \ (v_i \in V, \ k = 1, \ldots, m) \qquad \text{(VI.43)}$$

$$\sum_{k=1}^{m}(l_{ij}^k + l_{ji}^k) = \lceil \frac{q_{ij}}{Q} \rceil \quad \text{(VI.44)}$$

$$x_{ij}^k \geq l_{ij}^k \quad ((v_i, v_j) \in A,\ k = 1, \ldots, m) \quad \text{(VI.45)}$$

$$\sum_{(v_i,v_j)\in A} q_{ij} l_{ij}^k \leq Q \quad (k = 1, \ldots, m) \quad \text{(VI.46)}$$

$$\left.\begin{array}{l} \sum_{v_i,v_j \in S} x_{ij}^k \leq |S| - 1 + n^2 u_k^S \\ \sum_{v_i \in S} \sum_{v_j \notin S} x_{ij}^k \geq 1 - w_p^S \\ u_k^S + w_k^S \leq 1 \\ u_k^S,\ w_k^S \in \{0,1\} \end{array}\right\} (S \subseteq V \setminus \{v_1\};\ S \neq \emptyset;\ k = 1, \ldots, m) \quad \text{(VI.47)}$$

$$x_{ij}^k,\ l_{ij}^k \in \{0,1\} \quad (v_i, v_j) \in A;\ (k = 1, \ldots, m) \quad \text{(VI.48)}$$

Na formulação acima adaptada por [Eiselt et al 95ii], as restrição VI.43 consideram a conservação de fluxo para cada veículo. As restrições VI.44 consideram o serviço apenas naqueles arcos com demanda positiva. As restrições VI.45 definem que um arco é servido por um veículo se e somente se ele é atravessado pelo mesmo veículo. As restrições VI.46 garantem qua a capacidade de um veículo nunca é excedida. As restrições VI.47 garantem que a solução não conterá subrotas ilegais, ou seja, uma rota de um veículo só é legal se forma um circuito a partir do vértice origem (depósito central).

As restrições VI.47 funcionam do seguinte modo: Para um dado k e S, apenas uma das variáveis binárias u_k^S ou w_k^S podem assumir o valor 1. Qualquer ciclo poderá conter um conjunto de vértices S e arcos atravessados pelo veículo k, e deve ser conetado a $V \setminus S$ (como também a v_1), pois:

$$\sum_{v_i,v_j \in S} x_{ij}^k \geq x_{ij}^k \geq |S| - 1 \Rightarrow u_k^S = 1 \Rightarrow w_k^S = 0 \Rightarrow \sum_{v_i \in S} \sum_{v_j \notin S} x_{ij}^k \geq 1$$

O número de restrições desta formulação atinge a $O(m|V|^2)$, [Eiselt et al 95ii].

Na formulação não direcionada, proposta por [Belenguer & Benavent 91], as variáveis não direcionadas x_{ij}^k e l_{ij}^k estão definidas apenas para $i < j$. Além disso, x_{ij}^k representa o número de vezes que o elo (v_i, v_j) é atravessado pelo veículo k sem ser servido pelo veículo. Definindo $E(S) = \{(v_i, v_j) : v_i \in S, v_j \in V \setminus S, v_j \in S\}$ e $E^+(S) = E(S) \cap \{(v_i, v_j) \in E : q_{ij} > 0\}$. A formulação é a que segue.

(**PRAD** Não Orientado) Minimizar $\displaystyle\sum_{k=1}^{m} \sum_{(v_i,v_j)\in A} c_{ij}(x_{ij}^k + l_{ij}^k)$ (VI.49)

sujeito a,

$$\sum_{k=1}^{m} l_{ji}^k = 1, \ ((v_i, v_j) \in A \text{ e } q_{ij} > 0) \quad \text{(VI.50)}$$

$$\sum_{k=1}^{m} q_{ij}^k l_{ij}^k \leq Q \quad \text{(VI.51)}$$

$$\sum_{(v_i,v_j)\in E(S)} x_{ij}^k + \sum_{(v_i,v_j)\in E^+(S)} l_{ij}^k \geq 2l_{ht}^k, \quad \begin{array}{l}(S \subseteq V \setminus \{v_i\};\ S = \emptyset;\\ k = 1, \ldots, m;\ v_h, v_t \in S \text{ e } q_{kt} > 0)\end{array} \quad \text{(VI.52)}$$

$$\left.\begin{array}{l}\sum_{(v_i,v_j)\in E(S)} x_{ij}^k + \sum_{(v_i,v_j)\in E^+(S)} l_{ij}^k = 2z_k^S \\ z_k^S \in \mathcal{Z}_+\end{array}\right\} \begin{array}{l}(S \subseteq V \setminus \{v_1\};\\ S \neq \emptyset;\ k = 1, \ldots, m)\end{array} \quad \text{(VI.53)}$$

$$x_{ij}^k,\ l_{ij}^k \in \{0,1\} \ ((v_i, v_j) \in A);\ (k = 1, \ldots, m) \quad \text{(VI.54)}$$

As restrições VI.51 garantem que todo elo com demanda positiva é servido exatamente uma vez por um veículo. As restrições VI.52 limitam as rotas pela capacidade dos veículos. As restrições VI.53 são similares às restrições VI.46, elas garantem que se S contém um elo (v_h, v_l) servido pelo veículo k, então S deve ser ligado a seu complemento servido pelo mesmo veículo. As restrições VI.54 estipulam que qualquer vértice não vazio do conjunto S não contendo o depósito, deve estar conectado a seu complemento um número par de vezes por qualquer veículo. Estas restrições não podem ser expressadas através apenas das variáveis x_{ij}^k e l_{ij}^k. No entanto [Belenguer & Benavent 91] mostraram que as seguintes restrições são válidas e quase sempre suficientes para permitir a paridade.

$$-\sum_{(v_i,v_j)\in E(S)} x_{ij}^k + \sum_{(v_i,v_j)\in T} l_{ij}^k$$

$$-\sum_{(v_i,v_j)\in E^+(S)\setminus T} l_{ij}^k \leq |T| - 1 \ (S \subseteq (V - \{v_1\});$$

$$S \neq \emptyset;\ T \subseteq E^+(S); |T| \text{é impar})$$

As restrições acima são válidas desde que $\sum_{(v_i,v_j)\in T} l_{ij}^k \leq T$ e, se a igualdade se verifica,

$$\sum_{(v_i,v_j)\in E(S)} x_{ij}^k + \sum_{(v_i,v_j)\in E^+(S)\setminus T} l_{ij}^k$$

atinge pelo menos o valor 1 de modo a atingir a condição de paridade a ser satisfeita devido $|T|$ ser par.

Os autores usaram um método *Branch & Cut* semi-automático, onde atingiram a solução ótima para dois problemas, um com $|V| = 16$ e $|A| = 24$ e o outro com $|V| = 24$ e $|A| = 34$ em ambiente IBM PC-AT.

VI.3.2 Encontrando Limites Inferiores para o PRAD

A grande dificuldade deste problema está no conhecimento *a priori* dos seus limites inferiores. [Golden & Wong 81], [Assad et al 1987], [Pearn 88], [Win 87] assim como [Bonavent et al 92] desenvolveram a partir das formulações, limites inferiores para o PRAD. O trabalho de [Bonavent et al 92] inclui uma comparação teórica e computacional entre os limites inferiores conhecidos. Em todas as idéias, considera-se uma solução do problema de *1-matching* sobre um grafo H derivado de G.

1. Limites Inferiores de [Golden & Wong 81]

   Passo 1: Determinar o limite inferior do número de veículos, o qual é obtido de

   $$\underline{m} = \lceil \sum_{(v_i,v_j)\in A} \frac{q_{ij}}{Q} \rceil.$$

   No entanto, qualquer limite do problema de *binpacking* pode ser usado para este caso, [Martello & Toth 90].

   Passo 2: Seja d_i o grau do vértice v_i. Definir,

   $$\lambda = \begin{cases} 2\underline{m} - d_1, & \text{Se } d_1 \text{ é par} \\ 2\underline{m} - d_1 - 1, & \text{Se } d_1 \text{ é ímpar} \end{cases}$$

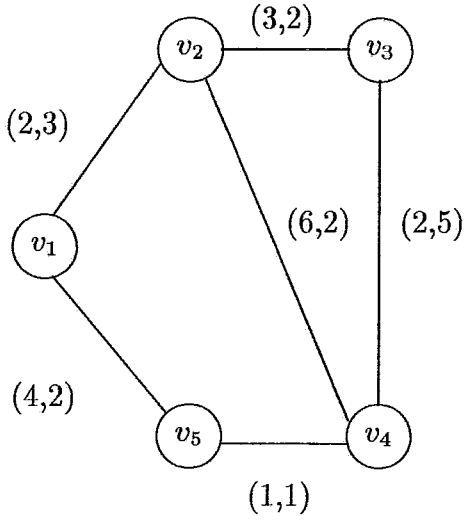
   Passo 3: (Grafo de Emparelhamento) Construir $H = (U, B)$ como segue.

   - O conjunto de vértices U é igual à união de três conjuntos disjuntos, T, K e P, onde $T = \{v_i \in V : d_i \text{ é ímpar}\}$ e $K = \{k_1, \ldots, k_\lambda\}$, $P = \{p_1, \ldots, p_\lambda\}$ são os conjuntos de vértices artificiais. Se $\lambda \leq 0$, então K e P são vazios.
   - Definir os arcos (u_r, v_s) de B e seus custos associados d_{rs}, do seguinte modo, figura VI.15:
     - Se $u_r, u_s \in T$, d_{rs} é o comprimento da menor cadeia em G entre $v_r \equiv u_r$ e $v_s \equiv u_s$;
     - Se $u_r \in T$ e $u_s \in K$, d_{rs} é o comprimento da menor cadeia em G entre $v_r \equiv u_r$ e v_1;
     - Se $u_r = k_t \in K$ e $u_s = l_t \in P$ para algum t, $d_{rs} = min_{j>1}\{c_{ij}\}$;
     - Se $u_r, u_s \in P$ então $d_{rs} = 0$.

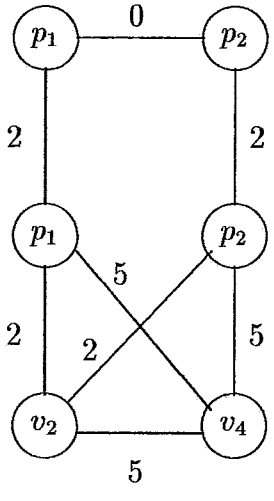
   Passo 4: (Limite Inferior) Aplicar o algoritmo de *1-matching* em H. Se $z(H)$ é o custo da solução do problema de *1-matching*, então:

   $$LI_0 = z(H) + \sum_{(v_i,v_j)\in A: q_{ij}>0} c_{ij},$$

   onde LI_0 é um limite inferior válido para o PRAD.



a. Grafo G com pares (c_{ij}, q_{ij}), $Q = 10$, $\underline{m} = 2$.



b. Grafo H com custos d_{rs}.

Figura VI.15: Transformação de um grafo G em um grafo H.

2. Limites Inferiores *Node Scanning*, [Assad et al 1987]

   Passo 1: Tomar de G o comprimento da menor cadeia entre v_i e v_j em G, e seja d_i o grau de v_i, como no LI de [Golden & Wong 81].

   Passo 2: Rotulando novamente todos os vértices na ordem ascendente de d_{1i}, seja $p = min\{k : \sum_{i=2}^{k} d_i \geq \lambda\}$, onde

$$\lambda = \lceil \frac{|A|}{\alpha^{-1}} \rceil, \ \alpha = min\{q_{ij}/Q\}.$$

   Fazer $d_p = \lambda - \sum_{i=2}^{p-1} d_i$.

   Passo 3:

$$LB_1 = \sum_{\substack{(v_i, v_j) \in A \\ q_{ij} > 0}} c_{ij} + \sum_{i=2}^{p} d_{1i} d_i$$

   , onde LI_1 é um limite inferior válido para o PRAD.

3. Limite Inferior de [Bonavent et al 92]

   Passo 0: Definir para todo subconjunto próprio S de V o conjunto de corte $E(S) = \{(v_i, v_j) \in A : v_i \in S, v_j \in V \setminus S$ ou $v_i \in V \setminus S, v_j \in S\}$. Seja T o subconjunto de todos os vértices ímpares ao longo de v_1 e os vértices de V incidentes a pelo menos um elo com demanda positiva.

   Passo 1: (Inicialização) Fazer $S := \{v_1\}$, $L_1 := L_2 := 0$, e

$$\bar{c} = \sum_{\substack{(v_i, v_j) \in A \\ q_{ij} > 0}} c_{ij}$$

   Passo 2: (Solução 1-Matching) Fazer $V' = V \setminus S$. Seja G um grafo induzido de V', identificar os componentes conexos de $G'_s = (V'_s, A'_s)$ de G' $(s = 1, \ldots, t)$

   Para $(s = 1, \ldots, t)$ fazer

$$q_s := |\{(v_i, v_j) \in E(V'_s) : q_{ij} > 0\}|$$

$$p_s := \lceil (\sum_{(v_i, v_j) \in A'_s \cup E(V'_s)} q_{ij})/Q) \rceil$$

$$r_s := max\{0, 2p_s - q_s\}$$

$$c_s := min_{(v_i, v_j) \in E(V'_s)}\{c_{ij}\}$$

   Fazer $T_s := T \cap V'_s$. Se $T_s \neq \emptyset$ ou $r_s > 0$ então construir um grafo valorado H_s (b), resolver o problema de *1-matching* sobre H_s. Com o valor da solução de $z(H_s)$, fazer $m_s := z(H_s)$.

Fazer $L_1 := max\{L_1, \bar{c} + L_2 + \sum_{s=1}^{t} m_s\}$ e $L_2 := L_2 + \sum_{s=1}^{t} r_s c'_s$.

Passo 3: (Atualizando o Conjunto de Vértices) Fazer $S' = \{v_i \in V : v_i$ é adjacente a um vértice de $S\}$ e $S := S \cup S'$. Se $S \neq V$, ir para o passo 2. Caso contrário, fazer $LI_2 := L_1$ e terminar.

O grafo H_s pode ser construido do seguinte modo:

Para $v_i \in V'_s$, seja c_i a distância mínima entre v_i e qualquer vértice de S, rotular novamente os vértices de V'_s como $v_{i_1}, v_{i_2}, \ldots, v_{i_k}$ tal que $c_{i_1} \leq c_{i_2} \leq \ldots \leq c_{i_k}$. Tomar h como o menor inteiro satisfazendo $\sum_{j=1}^{h} d_{ij} \geq r_s$, onde d_{i_j} é o grau do vértice v_{i_j}.

O conjunto de vértices de H_s é dado por $J \cup K \cup T'_s \cup P$, onde J é o conjunto de r_s vértices artificiais; k contém d_{ij} cópias do vértice $v_{ij} \in V'_s$ para $j = 1, \ldots, h$; T'_s contém uma cópia de cada vértice de T_s, exceto para aqueles vértices cujas cópias já foram incluidas em K; P é o conjunto de $max\{0, |T_s| - r_s\}$ variáveis artificiais.

Os arcos de H_s e seus custos são definidos como seguem: Qualquer arco de $K \cup T'_s$ tem o custo de um arco entre duas cópias do mesmo vértice assumindo o valor zero; se $v_k \in K \cup T'_s$ e $v_l \in J \cup P$, arcos (v_k, v_l) têm um custo igual a c_i, onde v_i é o vértice de V'_s correspondendo a v_k; se $v_l \in P$, o arco (v_k, v_l) tem custo zero.

Dentre os limites acima apresentados, LI_2 é teoricamente melhor que os demais, pois em geral há uma melhora em relação a LI_0 não somente no número de veículos necessários para cobrir o grafo, como também no número de veículos necessários para determinados subgrafos, [Bonavent et al 92].

VI.3.3 Métodos Heurísticos para o PRAD

A seguir detalha-se alguns dos métodos heurísticos propostos para o PRAD, no que concerne a sua evolução algoritmica. As classes de algoritmos provêm em geral dos métodos de Roteamento em Nós, são eles:

1. Construtivos Simples;
2. Construtivos de Duas-Fases;
3. De Melhorias;
4. De Agrupar-Primeiro Rotear-Depois;
5. De Rotear-Primeiro e Agrupar-Depois.

1. **Algoritmo Construir-Cancelar**, [Christofides 73] e [Pearn 89] - Complexidade: $O(|E|.|V|^3)$

   Passo 1: Construir ciclos viáveis (tendo uma demanda total nunca maior que a capacidade do veículo Q) os quais, quando removidos, não separam os grafos restantes em componentes desconexas. Quando os ciclos viáveis estão construidos, remover todos os elos nos ciclos do grafo. Repetir o ciclo de procedimento *Construir-Então-Remover* até que não haja mais ciclos (viáveis) a serem encontrados.

   Passo 2: Aplicar o algoritmo de *1-matching* para a geração de caminhos eulerianos e juntar os vértices de grau ímpar no grafo remanescente (com os ciclos construidos removidos), assim, um ciclo de Euler deve ser gerado pela adição da solução do *1-matching* ao grafo restante. O ciclo viável do procedimento *Construir-Então-Remover* e o procedimento de *1-matching* são repetidos até que todas as demandas dos arcos no grafo tenham sido cobertas.

2. **Algoritmo Path-Scanning**, [Golden et al 83] - Complexidade: $O(|V|^3)$

   Passo 0: $z_a := \infty$, $z_\star := \infty$.

   Passo 1: (Formando Ciclos) Extender um caminho pela adição de um elo que pareça mais favorável de acordo com o critério em uso, podendo ser um dos abaixo relacionados.

   - A relação c_{ij}/q_{ij}, onde se minimiza a demanda restante;
   - $max\{c_{1j}\}$;
   - $min\{c_{1j}\}$;
   - Se a capacidade restante do veículo é menor do que a metade, tomar o $max\{c_{1j}\}$, caso contrário $min\{c_{1j}\}$;

   Passo 2: Ir para o passo 1 até que não haja mais demandas a incluir. Fazer z_a igual ao valor da solução usando o critério assumido.

   Passo 3: Se $z_a < z_\star$ Então Armazenar esta solução como a melhor obtida. Fazer $z_a := \infty$ e mudar o critério. Ir para o passo 1 caso haja ainda critério a ser explorado.

3. **Algoritmo Aumentar-Unir**, [Golden et al 83] - Complexidade: $O(|V|^3)$

   Passo 1: (Inicializar) Todos os elos com demanda são servidos por ciclos separados;

   Passo 2: (Aumentar) Iniciando pelo ciclo de maior custo disponível, verificar se a demanda de um elo de um ciclo menor pode ser servida pelo ciclo maior.

Passo 3: (Unir) Considerando as restrições de capacidade, avaliar as ligações de quaisquer dois ciclos (é possível a inclusão de novas restrições). Juntar os dois ciclos que dão a maior economia.

Passo 4: (Iteração) Repetir o passo 3 até que não seja mais possível unir ciclos.

4. **Algoritmo Modificado Construir-Excluir**, [Pearn 89] - Complexidade: $O(|E|.|V|^4)$

   Passo 1: (Construir-Excluir 1) Iniciar com o vértice depósito e construir um ciclo viável. Para usar totalmente a capacidade do veículo, na formação de cada ciclo, o arco (i, j) adicionado (para extender ao caminho da rota) poderia maximizar a demanda total no caminho de menor demanda. Remover o ciclo do grafo. Repetir o procedimento *Construir-Então-Excluir* até que nenhum ciclo viável possa ser encontrado.

   Passo 2: (AGM) Se o grafo restante é desconexo, aplicar um algoritmo de AGM para fazer com que o grafo remanescente seja conexo, tratando cada componente conexa como um único vértice, onde a distância entre cada par de componentes é definida como sendo o comprimento do menor caminho que corresponde às ligações entre pares de componentes. Adicionar os elos da AGM ao grafo restante.

   Passo 3: (1-Matching) Aplicar o algoritmo de *1-matching* no grafo formado no passo 2, de modo que um ciclo euleriano seja formado. Se ciclos viáveis podem ser encontrados, retornar ao passo 1.

   Passo 4: (Construir-Excluir 2) Iniciar com o vértice depósito. Construir um ciclo pela adição de arcos com demanda arbitrária e extender o caminho até que a capacidade do veículo seja exaurida. Remover o ciclo e todos os arcos artificiais do grafo. Ir para o passo 2.

   Passo 5: Repetir os passos de 1-4 até que todo o grafo seja coberto.

   Passo 6: Considerar que o ciclo inicial a ser removido será um outro ciclo diferente daqueles já tomados. Se não houver mais ciclos a tomar, admitir a solução da heurística como aquela com o melhor ciclo de partida e terminar, caso contrário vá para o passo 2.

5. **Algoritmo Modificado Path-Scanning**, [Pearn 89] - Complexidade: $O(|V|^3)$

   A idéia aqui é apenas modificar o critério de escolha do seguinte modo:

   - Selecionar um dos cinco critérios aleatoriamente e usá-lo em cada passo. Gerar uma solução completa. As melhores k-soluções poderiam ser escolhidas como a solução final do método.

- Selecionar um dos cinco critérios com várias probabilidades (por exemplo, o critério 1 recebe probabilidade 0.6 e cada um dos outros quatro recebe probabilidade 0.1) a ser usado em cada passo, e gerar uma solução completa. As melhores k-soluções poderiam então ser escolhidas como uma solução final deste método.

Nos testes deste algoritmo, dentre várias composições de probabilidade para os critérios a escolhida foi $(0.2, \ldots, 0.2)$. Nota-se porém que a precisão deste algoritmo pode ser sempre melhorada à medida que se aumenta o número de soluções exploradas. Isto abre uma interessante fonte de pesquisa para algoritmos de busca metaheurísticos baseados em Busca TABU e Algoritmos Genéticos, principalmente porque este método possui como vantagem a propriedade de ter uma complexidade mais aceitável que os outros.

6. **Algoritmo Inserção-Paralela**, [Chapleau et al 84]

   Este algoritmo foi projetado especialmente para o contexto de roteamento de ônibus escolares, dentro dos critérios deste problema específico. Ele gera rotas em paralelo, considerando estratégias complementares conforme abaixo:

   (a) Dado uma ligação, determinar que ligação remanescente deve ser inserida a seguir.

   (b) Dada uma rota, determinar que ligação remanescente deverá ser inserida a seguir.

   Passo 1: Inicialmente definir o número de rotas, simplesmente dividindo o número de estudantes pela capacidade dos ônibus.

   Passo 2: Inicializar as rotas usando a estratégia (a). A seguir usar a estratégia (b) para preencher as rotas. Os procedimentos de trocas são aqueles propostos por [Lin 65], [Lin & Kerningham 73] e versões relaxadas destas estratégias (onde restrições de capacidade e comprimento podem ser um pouco violadas), são usadas com o objetivo de reduzir o comprimento total e iniciar novas rotas caso necessário.

   Passo 3: Balancear as rotas preenchendo as rotas iniciais vazias.

   Passo 4: Remover uma ou duas das piores rotas, reduzindo a distância total da viagem.

7. **Algoritmo Aumentar-Inserir**, [Pearn 91] - Complexidade $O(|V|^3)$

   É um procedimento composto de duas fases, Aumentar e Inserir, baseados nas técnicas de Aumentar-Unir e Inserção Paralela com algumas modificações. São métodos robustos e fáceis de serem implementados. Neste procedimento as rotas que atendem ligações distantes do depósito possuem um melhor planejamento com relação a seu serviço e os problemas com composição de demandas altas e tratamento em redes esparsas são mais fáceis de serem resolvidos.

Dada uma rede com depósito situado no vértice 1, define-se:

d_{ij}, custo do menor caminho do vértice i ao j.

$D_{ij} = d_{1i} + d_{1j}$.

$DCiclo_{ij}$, o ciclo de menor quantidade (com respeito à demanda das ligações) que cobre a ligação (i, j).

$INCusto_{uv}$, custo adicional que ocorre na inserção da ligação (u, v) no ciclo corrente do caminho.

S, é o conjunto de ligações com demanda de $G = (V, L)$ que não foram servidos.

- Fase I - AUMENTAR
  (a) Fazer $S = L$. Iniciar com a ligação $(i, j) \in S$ com demanda $D_{ij} = max_{u,v \in S}\{d_{u,v}\}$. Encontrar o ciclo $DCiclo_{ij}$, então proceder o passo 2. Nota-se que a busca de $DCiclo_{ij}$ deve ser repetida para todo $(u, v) \in S$, proceder fase II.
  (b) Aumentar o caminho do ciclo inicial (consistindo do arco ij) pela adição das ligações com demanda $(u, v) \in DCiclo_{ij}$ na ordem descendente de D_{uv} até que a capacidade do veículo seja exaurida. Os caminhos de retorno de menor custo dos vértices finais para extender o comprimento do ciclo ao depósito de vértice 1 é seguido para formar um ciclo completo. Remover este ciclo completo do grafo G e atualizar S.
  (c) Repetir os passos 1 e 2 da fase 1 até que nenhum $DCiclo_{ij}$ seja encontrado para todo $(i, j) \in S$. Proceder a Fase II.
- Fase II - INSERIR
  (a) Iniciar com a ligação $(i, j) \in S$ com $D_{ij} = max_{(u,v) \in S}\{D_{uv}\}$, inserir o arco de demanda $(u, v) \in S$ em um caminho de ciclo inicial (consistindo da ligação (i, j)), onde $INCusto_{uv} \leq \overline{c}$, onde $\overline{c}$ (a ser variado sobre um dado conjunto de valores) é um limite superior sobre o custo de inserção.
  (b) Repetir este processo para todo $(i, j) \in S$ na ordem decrescente de D_{uv} até que a capacidade do veículo seja exaurida ou o vértice depósito 1 seja encontrado. Os caminhos de retorno de custo mínimo dos dois pontos finais do caminho do ciclo extendido para o depósito são seguidos de um ciclo completo. Promover este ciclo completo do grafo G e atualizar S.
  (c) Repetir os passos 1 e 2 da Fase II até que todo grafo seja coberto.

Escolha do Limite Superior $\overline{c}$

O limite inferior $\bar{c}$ para o custo de inserção $INCusto_{uv}$ não deve ser muito alto ou muito baixo. Se $\bar{c}$ é muito alto, o procedimento tenta inserir ligações com demanda que estão bem afastadas do caminho atual, gerando grandes ciclos. Por outro lado, se $\bar{c}$ é admitido ser muito baixo, o procedimento apenas insere arcos com demanda os quais estão *muito próximos* do ciclo corrente; consequentemente, o caminho do ciclo poderia atingir o depósito muito antes que a capacidade do veículo seja esgotada. Portanto, muitos ciclos adicionais devem ser gerados de modo a cobrir todo o grafo. Os autores não encontraram regras gerais para a determinação de $\bar{c}$, no entanto, eles definiram como valor razoável o intervalo $[0, max_{i \in V}\{d_{1i}\}]$.

No cálculo de $INCusto_{uv}$ os autores extenderam o método de economias de [Clarke & Write 64] para incluir ambos os finais dos caminhos dos arcos e ciclos; isto é, para um ciclo com pontos finais i, j e demanda das ligações (u, v),

$$INCusto_{uv} = min_{(x,y) \in B}\{d_{ix} + d_{1y} - d_{1i},\ d_{jx} + d_{1y} - d_{1j}\}$$

, onde $B = \{(u, v), (v, u)\}$.

Entre as estratégias usadas pelos autores na confecção dos algoritmos Aumentar-Inserir, um dos métodos usa o ciclo de menor custo $DCiclo_{ij}$ e o outro usa a menor quantidade $QCiclo_{ij}$.

Os autores mostram para um conjunto de casos testes, $13 \leq |V| \leq 27$ e $23 \leq |L| \leq 51$ os algoritmos de Inserir-Aumentar atingiram em grande parte os melhores resultados, porém algumas instâncias atingiram em seus piores valores resultados acima de 74% distantes dos limites inferiores propostos.

8. **Algoritmos Agrupar-Primeiro Rotear-Depois** - [Win 87] e [Bonavent et al 90]

   Nesta classe de algoritmos, os arcos são particionados em grupos, cada um tendo um peso total não execedendo a Q. Um critério guloso é então aplicado [Win 87] ou um algoritmo de atribuição generalizado [Bonavent et al 90] é usado. Uma rota de veículo é determinada para cada grupo por uma modificação simples no algoritmo do PCC.

9. **Algoritmo Rotear-Primeiro Agrupar-Depois** - [Win 87]

   Passo 1: Construir uma grande rota euleriana passando por todas as ligações com demanda positiva. Se todas as ligações induzem um grafo conexo, então esta rota pode ser obtida em tempo polinomial por simples modificação do PCC. Caso contrário o problema é um PCR, e pode-se aplicar os algoritmos aproximativos de [Frederickson 79].

   Passo 2: Particionar a grande rota em grupos viáveis, ou através de técnicas de empacotamento (*NextFit*), [Win 87], onde para cada grupo formado se tem uma rota de um veículo.

Dentre as heurísticas acima descritas com detalhes, a literatura não desenvolveu como no caso dos PRV em nós, problemas testes padrões que verificassem os seus desempenhos, uma vez que muitas delas foram desenvolvidas com o objetivo de propor soluções a problemas muito específicos. Em [Pearn 89] e [Pearn 91], tem-se os trabalhos mais criteriosos nesse sentido, uma vez que avalia com propriedade um conjunto de casos testes entre as heurísticas mais usadas. Os testes indicaram que sobre grafos densos, o algoritmo modificado de Construir-Excluir atinge as melhores soluções em problemas testes com $11 \leq |V| \leq 17$, e densidade de arcos entre 70% e 100%; atingindo sempre uma solução ótima. Já para grafos esparsos com demanda nas ligações relativamente alta, o algoritmo Aumentar-Inserir mostra-se como o melhor, como foi visto.

As heurísticas acima apresentadas usam fortemente as técnicas sobre arcos explorando sempre algoritmos de *1-matching*, árvores geradoras e/ou caminhos mínimos. As transformações dos grafos são sempre baseadas nos próprios arcos, elemento que define a corrente de pensamento da literatura desde a formulação do problema por [Christofides 73] e [Golden & Wong 81]. Uma abstração desta corrente de pensamento será discutida no próximo capítulo, onde avalia-se a abordagem da transformação de arcos em nós para a solução do PRAD, e mais adiante para o encaminhamento de soluções heurísticas de boa qualidade a qualquer problema de percurso de veículos seja em arcos seja em nós, considerando sua formulação básica, ou o PPVC (Problema de Percurso de Veículos Capacitados).

Capítulo VII

Transformações de Arcos em Nós

A primeira vista a idéia de resolver problemas de percursos de veículos que estão relacionados ao PGR extrapola a curiosidade da solução através das formulações de arcos, devido à disponibilidade dos algoritmos exatos de 1-*matching* de custo mínimo, de árvores geradoras mínimas ou de 2-*matching*. O uso de arcos como elemento de transformação dos nós (em cadeia de nós), reduz o problema geral de percurso de forma considerável, [Orloff 74].

Apesar de [Orloff 74] ter desenvolvido um trabalho exclusivamente voltado para transformação de nós em arcos, tendo mostrado que qualquer PCV poderia ser transformado em um equivalente problema de percurso em arcos, neste capítulo tem-se a preocupação de mostrar que as transformações inversas (arcos em nós) podem acontecer de modo tão razoável quanto a primeira, e com boa qualidade nas soluções.

Uma argumentação relevante sobre esse assunto refere-se a que tanto o PCV quanto o PCR comportam-se como problemas nitidamente $\mathcal{NP}$-Árduos na grande maioria das aplicações. As formulações do PRV e do PRAD ajudam a encontrar soluções exatas, como foi visto nos capítulos V e VI, para problemas apenas modestos. Em geral, as heurísticas de percurso em arcos com demanda ressentem-se por sua estabilidade, podendo desviar-se da solução ótima a um valor proibitivo. Já nos problemas em nós, as heurísticas têm em geral um comportamento mais estável quanto à proximidade da solução ótima, com resultados significativamente interessantes, [Laporte 92], [Bonavent et al 92], [Campello & Maculan 94].

A seguir mostra-se algumas das transformações propostas pela literatura, e discute-se sua propriedade no que se refere à expansão da rede e no tratamento das instâncias a que se aplicam. Propõe-se também novos métodos de transformação, transformação por quebra de arcos e transformação linha, que mostraram-se aplicáveis a um grande espectro de problemas de percurso de veículos.

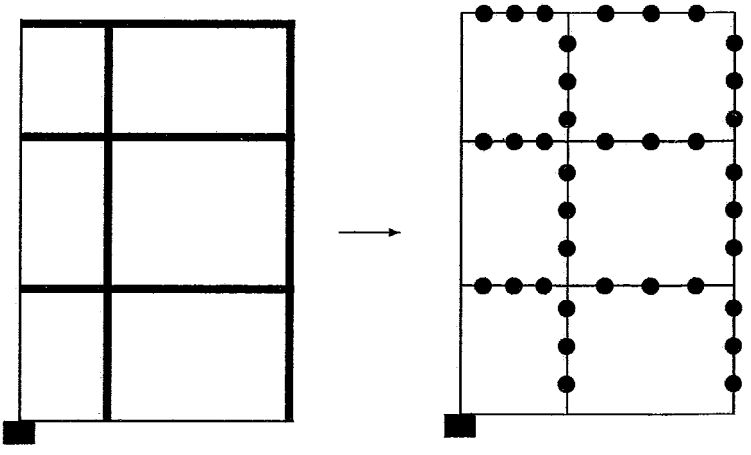


Figura VII.1: Transformação de elos com demanda em nós, pela inclusão na rede de $3|R|$ nós em um grafo.

VII.1 Transformação de Pearn, Assad & Golden

As primeiras idéias de transformações de arcos em nós surgiram com o trabalho de [Pearn et al 87], onde se mostrou a possibilidade da quebra de uma rede através dos seus elos com demanda, em quatro partes iguais introduzindo três novos vértices na rede, cada um com um terço da demanda do elo original. Uma vez garantindo que os vértices de um mesmo elo com demanda seriam atendidos por um único veículo, a idéia proposta seria de aplicar um algoritmo de PRV sobre os vértices com demanda da rede expandida. Os autores garantiram que a transformação proposta se aplicaria bem ao caso do PRAD. Supondo R o conjunto de elos com demanda de um grafo $G = (V, E)$, $R \subseteq E$, $|R|$ é número de elos com demanda. Na figura VII.1 vê-se o processo de transformação proposto.

[Pearn et al 87] consideraram a transformação para um grafo não orientado. Na verdade, ela apenas se verifica quando se considera elos que estão em ciclos. Naqueles casos onde um elo não forma um ciclo, um retorno em U acontecerá sem que se calcule diretamente a solução através dos algoritmos do PRV. Nesta situação, sugere-se a colocação de um único vértice com demanda no vértice terminal, figura VII.2. Já para a situação onde se tem apenas arcos orientados, um único vértice poderá se posicionar no meio do arco de forma a representá-lo.

[Pearn et al 87] não mostraram nenhuma avaliação computacional do procedimento proposto, porém concluiram seu trabalho colocando sobre a dificuldade computacional inerente ao problema em virtude da inclusão de um número tão significativo de vértices com demanda na rede. Também consideraram que a avaliação computacional do problema seria uma forma interessante de checar se a

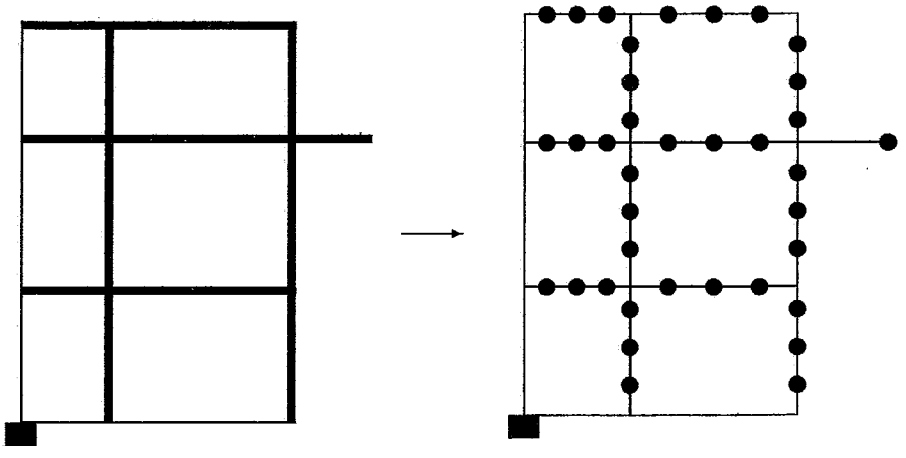


Figura VII.2: Revisão da proposta de Pearn, Assad & Golden no caso de um elo sem demanda que não pertence a ciclos do grafo.

transformação proposta de um problema de arcos e resolvê-lo com a transformação através de um algoritmo exato de roteamento em nós possui uma eficiência computacional valiosa.

De fato, tomando-se as aplicações reais de roteamento em arcos por mim desenvolvidas, um único veículo pode atender numa única rota cerca de sessenta a oitenta arcos, enquanto que numa aplicação de roteamento em nós normalmente uma rota dificilmente ultrapassa os cinquenta nós. Em virtude disto, todo o esforço empreendido no problema em nós deve-se somar à elaboração de procedimentos mais eficazes relativos à fase do PCV, no refinamento das rotas.

VII.2 Transformação de Ulosoy

A evolução das idéias de transformação continuaram com [Ulosoy 85]. Com um trabalho bastante refinado sobre um PRAD estendido, que considera basicamente um grafo qualquer $G = (V, L)$, uma frota heterogênea e a possibilidade da inserção de outras restrições como distância máxima a percorrer e tempo de serviço. A característica principal do método é a sua forma construtiva, baseada na técnica de rotear primeiro e agrupar depois. As fases do algoritmo de [Ulosoy 85] são descritas a seguir.

Algoritmo Geral de Ulosoy

Passo 1: Encontrar um grafo euleriano sobre G, e tomar uma *Grande Rota* euleriana, a partir do depósito central;

Passo 2: Particionar a *Grande Rota* em subrotas, satisfazendo a demanda dos veículos definidos;

Passo 3: Selecionar o conjunto de menor custo, satisfazendo a demanda dos veículos definidos;

Passo 4: Repetir os passos acima até que algum critério de parada seja satisfeito.

O passo de particionamento da *Grande Rota* pode ser melhor entendido como segue.

Particionamento da *Grande Rota*

Transformar $G = (V, L)$ em um G^*, onde $G^* = (V', A)$ é definido por:

1. V', é um conjunto de vértices;
2. A, é um conjunto de arcos;
3. $d(i)$, $\forall\ i \in V$ é a relação de incidência sobre os vértices;
4. $c'_{ij}, \forall < i, j > \in A$ é a estrutura de custos sobre os arcos.

Reduzindo inicialmente a rede G_E têm-se as seguintes regras:

Regra 1: Se houver alguma cadeia de arcos que corresponda apenas a uma travessia de arcos sem serviço na rota de Euler, então reduzí-la a um único arco com vértices iniciais e finais sendo aqueles da cadeia, sendo o seu custo igual o custo total da cadeia.

As regras seguintes mostram como deve ser construída a rede G^* a partir da rota euleriana encontrada.

Regra 2: Cada vértice de G^* corresponde a um caso na rota, exceto o primeiro vértice. Este, em G^*, corresponde ao vértice de depósito (vértice 1) de G. Os arcos na rota euleriana são rotulados como vértices de G^* na mesma ordem em que aparecem na rota, ou seja: o primeiro arco da rota euleriana corresponde ao vértice 2 em G^*, o segundo ao vértice 3 em G^* e assim por diante. Se o

último vértice de $G^\star$ representar um arco não servido, então ele é eliminado do conjunto dos nós.

Cada arco de $G^\star$ corresponde a uma rota viável em G, ou seja, um arco $< k, h >$ em $G^\star$ corresponde a uma rota do veículo em G a qual inclui todos as ligações representadas pelos vértices $(k+1), \ldots, h$ de $G^\star$.

Regra 3: Um arco de $G^\star$ é sempre formado de um vértice h orientado a um vértice k, onde $h < k$, $\forall\ h, k \in V'$, desde que a rota de um veículo correspondente não viole quaisquer restrições laterais (demanda, tempo, etc.) definidas no problema.

A presença dos arcos sem demanda na rota euleriana em G pode ser explorada para reduzir o número de arcos em $G^\star$, como determinado pela regra 3. Neste caso, se um vértice $i \in V'$ de $G^\star$ corresponder a um arco com demanda nula em G, as seguintes regras são adequadas para tratar este caso.

Regra 4: Nenhum arco de $G^\star$ entra no vértice i em $G^\star$, exceto o arco $< i-1, i >$.

Regra 5: Nenhum arco de $G^\star$ sai do vértice $(i-1)$ em $G^\star$, exceto o arco $< i-1, i >$.

As duas últimas regras foram introduzidas com o objetivo de prevenir que os arcos sem demanda estejam presentes nas rotas dos veículos sem propósito, e assim levando a rotas que venham a atingir soluções ruins.

Quando se particiona uma rota euleriana, obtêm-se uma sequência de arcos da rota euleriana que serão transformados em rotas dos veículos representadas em $G^\star$. Com este propósito, destinguem-se dois casos:

Caso 1: Os arcos na sequência tomada formam um ciclo.

(S1.) Se o vértice 1 de $G^\star$ é um membro da sequência, então a sequência corresponde à rota do veículo.

(S2.) Se o vértice 1 de $G^\star$ não é um membro da sequência, então conectar o vértice 1 a um vértice do ciclo que seja de menor custo.

Caso 2: Os arcos na sequência não formam um ciclo.

(S3.) Se o vértice 1 de $G^\star$ é um membro da sequência, então conectar o vértice 1 a dois vértices de grau ímpar usando a cadeia de menor custo.

(S4.) Se o vértice 1 de $G^\star$ não é um membro da sequência, então conectar o vértice a dois vértices de grau ímpar, usando a cadeia de menor custo.

Exemplificando melhor a construção do problema, considerando-se a rede da figura VII.3.

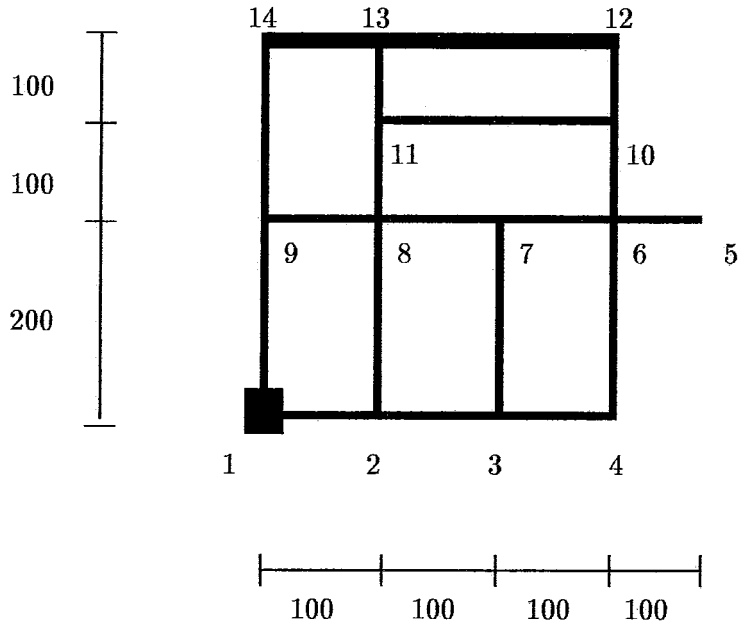


Figura VII.3: Rede de serviço onde $q_{ij} \propto ij$ para todo $ij \in L$.

Grande Rota euleriana = { 1, 2, 3, 4, 6, (5,6), 10, 12, 13, 14, 9, 8, 7, 6, 10, 11, (13, 11), (8, 7), (3, 2), (8, 9), 1 } onde (x,y) significa que são arcos de passagem sem serviço (*deadheading*), ou seja, (5,6), (13,11), (8,7), (3,2) e (8,9) são arcos que não serão rotulados na rede G^*.

Formando G^*, supondo que todo arco de G tem demanda proporcional à sua distância e que a frota é homogênea, cujos veículos possuem capacidade igual a 1000. O custo dos arcos da rede G^* será dado pelas soma dos custos dos arcos (c_{ij}) incluídos na rota do veículo e o custo fixo (CF) do veículo mínimo que atende aos arcos, figura VII.5.

$$\overline{c}_{2,7} = c_{1,2} + c_{2,6} + c_{6,10} + c_{10,1} + CF$$

No cálculo do caminho mínimo de 1 a m, onde m é o vértice formado pelo elo (9,1), tem-se que as rotas encontradas neste percurso definem a solução deste problema. Gerar quantas rotas eulerianas sejam necessárias de modo a encontrar aquele conjunto cuja solução seja a menor possível.

Alguns comentários são importantes sobre os passos do algoritmo proposto por Ulusoy, o primeiro é sobre a formação das rotas do PRAD. O fato de se considerar o custo fixo dos veículos, interfere sobremaneira no custo de formação das rotas, uma vez que o processo de minimização foi estabelecido como global, frota e distância total percorrida. Já no caso onde se deseja encontrar o grupo de rotas de custo mínimo, certamente que a preocupação com os veículos é irrelevante.

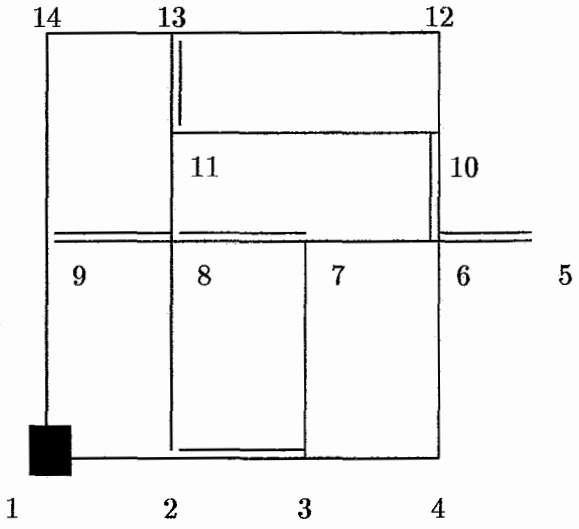


Figura VII.4: Rede euleriana mínima aplicando-se 1-*Matching*.

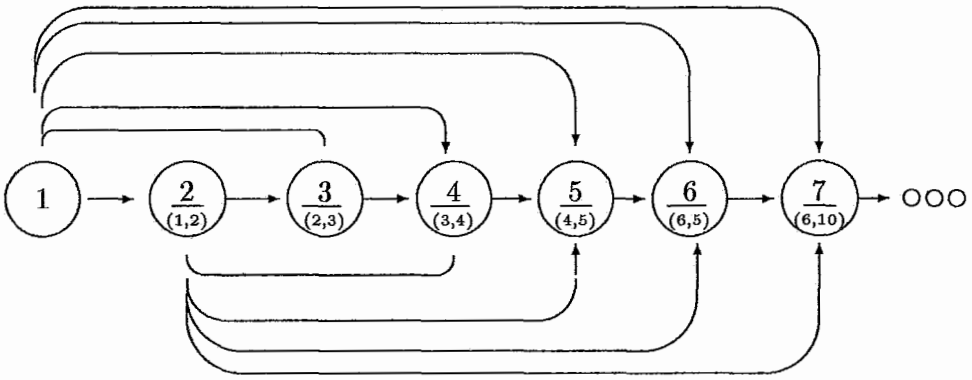


Figura VII.5: Transformação de Ulosoy.

O mérito maior nesta abordagem é então colocado sobre a facilidade ou não de formar grupos de rotas utilizando um algoritmo exato de caminho mínimo, além de se ter uma redução do problema a uma forma bastante simples e direta. Por outro lado, uma vez que muitas *Grandes Rotas* eulerianas podem ser formadas, e a idéia é conhecer aquela(s) que atinge(m) o menor custo, isto se torna uma desvantagem real, caracterizando fortemente a dificuldade computacional do problema.

Um procedimento de pós-processamento é sugerido pelo autor com o objetivo de formar rotas de custo total inferior. A idéia é baseada na troca de arcos sem serviço e com serviço ao longo das rotas dos veículos para a utilização mais eficiente das capacidades dos veículos. O procedimento é o que segue:

Algoritmo de Pós-Processamento de Ulosoy

Passo 1: Listar todos os arcos servidos que retornam pelo mesmo caminho sem serviço, na ordem decrescente das demandas. Ignorar aqueles arcos aos quais os únicos arcos que retornam pelo mesmo caminho ocorrem na mesma rota.

Passo 2: Escolher o arco com o maior serviço. Determinar se a deleção desta demanda da rota do veículo reduzirá as necessidades de capacidade para aquele veículo a níveis mais baixos. Se isto acontecer, ir ao passo 3. Senão, verificar se existe uma combinação de arcos da lista naquela rota do veículo a qual, quando excluída reduzirá as necessidades de capacidade para aquela rota de veículo a níveis mais baixos. Se isto ocorre, pegar aquela combinação que resulta numa utilização melhor da capacidade para o veículo, ir ao passo 3.

Passo 3: Considerar todas as rotas dos veículos onde este arco é um caminho de volta com um arco sem serviço. Determinar a troca que provoca a maior redução no custo e fazer as mudanças necessárias.

Se há mais de uma alternativa que provoca a mesma redução no custo, escolher aquela que resulta na máxima utilização da capacidade. Se nenhuma troca provoca melhoria nos custos, então não se faz nenhuma mudança. Eliminar o arco da lista e retornar ao passo 2 se a lista não está vazia, caso contrário parar. Se se vem do passo 2 com uma combinação de arcos, este passo é aplicado de acordo.

É importante lembrar que restrições adicionais, tais como tempo máximo de operação, número de paradas, etc. também poderiam ser consideradas no procedimento de trocas.

Em [Negreiros 90], mostra-se através de um exemplo que a *Grande Rota* pode ser formada a partir de critérios de formação de uma rota pela redução

dos esforços de movimentação propostos por [Bodin & Krush 78] e pouco depois por [McBridge 82]. Nos critérios adotados, a alternância entre servir e não-servir é considerada com custo adicional aos retornos em U, dobrar à direita, à esquerda ou seguir em frente. Na formação da rota de menor esforço a partir de uma rota euleriana previamente direcionada, os resultados obtidos com o uso da técnica de Ulosoy tornam-se superiores aos atingidos, inclusive considerando o pós-processamento. Isto se deve à formação da *Grande Rota* euleriana, que busca sempre formar percursos onde o serviço seja contínuo e com o menor número de alternâncias.

Na seção VII.5, discute-se mais detalhes sobre a formação de rotas com esforço mínimo, a partir das idéias propostas por [Bodin & Krush 78] e [McBridge 82], e uma nova formulação que se propõe para o problema através de uma redução ao PCV.

VII.3 Transformação por Quebra Simples de Ligações (QSL)

A idéia geral proposta por [Negreiros & Maculan 92a], surgiu da necessidade de se resolver o problema da formação de rotas econômicas para as sub-regiões de coleta de lixo na cidade de Fortaleza/Ce. O problema básico consiste de um grafo onde há demanda em nós e em ligações de uma rede mista. Além disto, restrições de tempo, locais de início e fim de cada viagem (podendo ser distintos) e restrições de tráfego podem ser consideradas na estrutura geral do problema, figura VII.6.

O Problema do Caminhão Coletor de Lixo Domiciliar (PCCLD) foi modelado pelos autores pressupondo que toda a rede de serviço seria formada por ligações com demanda, e veículos de igual capacidade. A formulação geral pode ser colocada como segue, [Negreiros & Maculan 92a].

Seja o seguinte conjunto de variáveis dadas por:

- nvg : Número de viagens do caminhão coletor;
- Q : Carga nominal associada à capacidade do veículo;
- T : Tempo limite de serviço;
- t_{ij} : Tempo de atravessar o arco/elo sem serví-lo (*deadheading*);
- t'_{ij} : Tempo de servir a ligação;
- $TP(s)$: Tempo de descarga no depósito para s-ésima viagem;

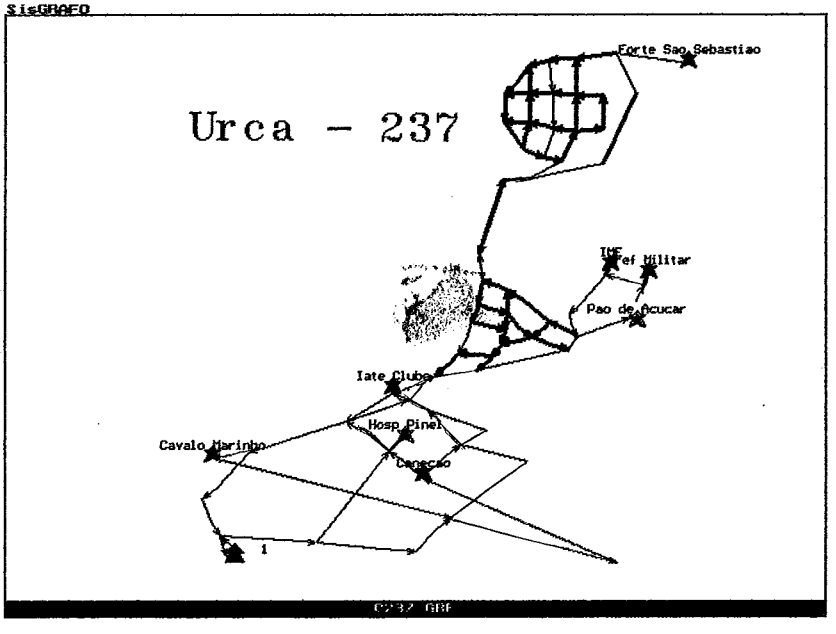


Figura VII.6: Situação de coleta de lixo domiciliar sobre ruas e pontos especiais no bairro da URCA, Rio de Janeiro.

- $(i, D)/iD$: Ligação direta que sai de qualquer vértice i e chega em D considerado como o vértice destino final;
- $(i, G)/iG$: Ligação direta que sai de qualquer vértice i e chega em G considerado como o vértice garagem;
- x_{ij}^s : Número de vezes que o veículo passa pela ligação ij na s-ésima viagem;
- $l_{ij}^s = \begin{cases} 1, & \text{Se o veículo passa coletando na ligação sobre a } s\text{-ésima viagem} \\ 0, & \text{Caso contrário} \end{cases}$
- $|S|$: Número de ligações no conjunto S;
- n : Número de vértices na rede de ruas;
- $\mathcal{N}$: Conjunto de vértices da rede;
- E : Conjunto de Elos na rede de ruas (segmentos de ruas de mão dupla);
- A : Conjunto de arcos na rede de ruas (segmentos de ruas de mão única);
- $A(i) = \{j \in N \mid (i,j) \in S\}$;
- $B(i) = \{j \in N \mid (j,i) \in S\}$;
- $S = E \cup A$: Conjunto de ligações da rede;
- A_1 : Conjunto de arcos da rede com demanda;
- E_1 : Conjunto de elos da rede com demanda;
- $S_1 \subseteq S$: Conjunto de ligações com demanda;
- $S_1 = E_1 \cup A_1$;

$$\textbf{(PCCLD)} \qquad \text{Minimizar} \quad z \tag{VII.1}$$

sujeito a:

$$\sum_{(i,j)\in S} \sum_{s=1}^{nvg} [x_{ij}^s \, t_{ij} + l_{ij}^s \, t'_{ij}] + \sum_{i\in B(D)} \sum_{s=1}^{nvg} x_{iD}^s \, TP(s) = z \tag{VII.2}$$

$$\sum_{j\in A(i)} x_{ij}^s = \sum_{j\in B(i)} x_{ji}^s \quad i \in \mathcal{N} - \{G, D\}, \ (s = 1, \ldots, nvg) \tag{VII.3}$$

$$\sum_{j\in A(G)} x_{Gj}^s = 1 \quad (s = 1, \ldots, nvg) \tag{VII.4}$$

$$\sum_{j \in B(D)} x^s_{jD} = \bar{l}^s \quad (s = 1, \ldots, nvg) \tag{VII.5}$$

$$\bar{l}^s \geq l^s_{ij} \quad (i,j) \in A_1, \ (s = 1, \ldots, nvg) \tag{VII.6}$$

$$\bar{l}^s \geq l^s_{ij} + l^s_{ji} \quad (i,j) \in E_1, \ (s = 1, \ldots, nvg) \tag{VII.7}$$

$$\sum_{j \in A(D)} x^s_{Dj} - \sum_{j \in B(D)} x^s_{jD} = 0 \quad (s = 2, \ldots, nvg) \tag{VII.8}$$

$$\sum_{s=1}^{nvg} (l^s_{ij} + l^s_{ji}) = 1 \quad (i,j) \in S_1 \tag{VII.9}$$

$$l^s_{ij} \leq x^s_{ij} \quad (i,j) \in S_1, \ (s = 1, \ldots, nvg) \tag{VII.10}$$

$$\sum_{(i,j) \in S_1} q_{ij} \, l^s_{ij} \leq Q \quad (s = 1, \ldots, nvg) \tag{VII.11}$$

$$\sum_{j \in A(i)} f^s_{ij} - \sum_{j \in B(i)} f^s_{ji} = \sum_{j \in A(i)} l^s_{ij} \quad i \in \mathcal{N} - \{D\}, \ (s = 1, \ldots, nvg) \tag{VII.12}$$

$$\sum_{j \in A(D)} f^s_{Dj} - \sum_{j \in B(D)} f^s_{jD} = - \sum_{(i,j) \in S_1} l^s_{ij} \quad (s = 1, \ldots, nvg) \tag{VII.13}$$

$$f^s_{ij} \leq |S_1| \, x^s_{ij} \quad (i,j) \in S, \ (s = 1, \ldots, nvg) \tag{VII.14}$$

$$f^s_{ij} \geq 0 \quad (i,j) \in S, \ (s = 1, \ldots, nvg) \tag{VII.15}$$

$$l^s_{ij} \in \{0,1\}, \ x^s_{ij} \in \mathcal{Z}^+ \quad (i,j) \in S, \ (s = 1, \ldots, nvg) \tag{VII.16}$$

As restrições VII.2 limitam o tempo de serviço total ao mínimo que o veículo possa fazer no atendimento da demanda total do circuito. As equações de VII.3-VII.8 são condições que garantem que o número de passagens por um dado vértice é par menos nos vértices relativos à Garagem e ao Destino Final. As restrições VII.9 garantem o serviço em uma única direção, em uma única viagem para uma ligação de S_1. As restrições VII.10 garantem que o veículo passa na s-ésima viagem servindo ou não uma dada ligação da rede de ruas. As restrições VII.11 guarantem a possibilidade de carregar o veículo servindo uma ligação de i para j, respeitando seus limites de carregamento. As restrições VII.12-VII.15 são as restrições propostas por [Golden & Wong 81] para evitar a formação de sub-rotas desconexas inviáveis as quais foram adaptadas ao **PCCLD**. As restrições VII.16 são as variáveis do modelo.

Assim como no PRAD, o PCCLD é um problema que de sua formulação tem O($nvg \ n^2$) desigualdades, com $nvg \ n^2$ variáveis {0,1} e 2 $nvg \ n^2$ variáveis inteiras. O PCCLD é um problema $\mathcal{NP}$-Árduo, pois pode-se transformá-lo diretamente num PRAD e daí num PCR. Este fato, como argumentam os autores é suficiente para se considerar a possibilidade da sua resolução através de heurísticas.

O modelo proposto para o PCCLD pode incorporar vértices com demanda, desde que se transforme um vértice em dois e um elo com custo zero e a capacidade do vértice. Esta expansão permite o tratamento do problema sem

prejuízo no cálculo da solução.

Dentre os diversos métodos em arcos que foram apresentados no capítulo VI, a idéia de formar um caminho euleriano em um subgrafo não foi considerada, mesmo sabendo que pequenas mudanças podem ser feitas na estrutura de formação do caminho euleriano a partir do ciclo euleriano. Além disto, não se pode considerar trivial a aplicação de qualquer procedimento sobre um grafo misto, para o PCR. Apesar da dificuldade encontrada, os algoritmos heurísticos para o PCV assimétrico podem tratar o problema do caminho hamiltoniano de custo mínimo com soluções sempre razoáveis para uma quantidade expressiva de vértices.

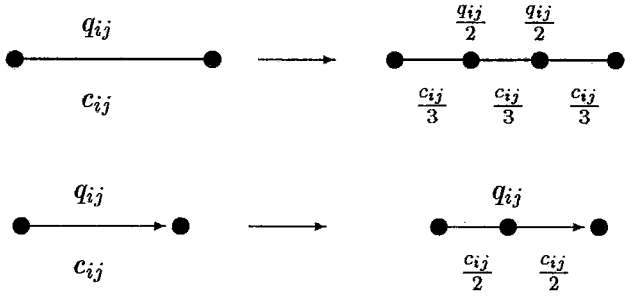


Figura VII.7: Quebra da rede proposta inicialmente por Negreiros 90.

Uma vez que os algoritmos sobre nós são mais diretos no tratamento geral do PCCLD, [Negreiros 90] optou inicialmente pela idéia da transformação proposta por Pearn, Assad & Golden. Algumas mudanças fundamentais foram introduzidas no procedimento geral de quebra, uma vez que na proposta deles havia a introdução de um grande número de vértices. A idéia inicial era de reduzir ao máximo o número de vértices explorados diferentemente na quebra de elos e arcos. Nos elos, dividiu-se a demanda ao meio e foi introduzido dois novos vértices os quais dividiam o elo em três novos elos com comprimento de um terço do elo original. Enquanto os arcos foram quebrados ao meio, onde se introduziu um único vértice com sua demanda total, figura VII.7.

Na fase de evolução do algoritmo MARCOLIX, uma vez que um dos vértices de um elo é atingido o seguinte também é obrigado a estar na mesma rota, caso contrário o elo é descartado. Além disso a travessia completa do elo era garantida através de marcadores de movimentação entre predecessores e sucessores dos elos e arcos, [Negreiros 90].

[Negreiros & Maculan 92a] no entanto evoluiram a idéia inicial, dispensando a quebra de elos em três partes, e consideraram apenas a quebra de elos em duas partes, concentrando a demanda total do elo no vértice correspondente da nova rede, uma vez que havia a garantia da travessia do elo de um lado a outro a partir dos vértices da rede original, figura VII.8.

O método proposto foi denominado como um procedimento de roteamento por rotulação de vértices, ou *Labeling Node Routing*, devido ao fato de que quando uma ligação é atingida o procedimento sinaliza que ela deve ser coberta do seu predecessor ao seu sucessor, devendo ser diferentes.

Considerando sobre a complexidade da quebra da rede, supondo-se uma rede mista $G = (V, L)$, onde $L = (E, A)$, e os conjuntos $V_d \subseteq V$, $E_d \subseteq E$ e $A_d \subseteq A$ relativos aos vértices, elos e arcos com demanda respectivamente, o novo grafo G' formado pela transformação proposta será $G' = (V', E', A')$ onde,

$$|V'| = |V| + |E_d| + |A_d|$$

$$|E'| = |E| + |E_d|$$

$$|A'| = |A| + |A_d|$$

que significa uma transformação linear, a qual pode ser processada em tempo linear, ou $O(|V'| + |L'|)$.

Figura VII.8: Quebra da rede (em elos) proposta por Negreiros & Maculan 92.

Um outro instrumento fundamental no algoritmo MARCOLIX, é a formação e o uso de uma estrutura de dados especial biconectada. Duas árvores **PiHp**, fila de prioridades, e **Pi1**, árvore binária balanceada, sendo a primeira ordenada pelo critério de distâncias e outra ordenada pelos rótulos dos vértices da rede aumentada com demanda, operam conjuntamente em todo o procedimento. Uma vez retirando um elemento de **PiHp** (topo) significa que o mesmo elemento em **Pi1** deverá ser retirado, porém sua posição pode não estar no topo de **Pi1**. Estas pilhas são usadas nos passos de verificação dos próximos rótulos a serem considerados a partir de uma lista de sucessores do último vértice incluido na rota.

O Algoritmo MARCOLIX

Passo 1 (Quebra de G): Dada uma rede G, identificar os vértices, arcos e elos com demanda, $q_i > 0$ ou $q_{ij} > 0$. Quebrar a rede usando o critério da figura VII.8, que serve para elos/arcos.

Passo 2 (Distância dos Rótulos): Determinar as distâncias aos vértices (rótulos) i a serem visitados (em serviço) nas rotas como segue.

$$\begin{cases} d'_i := d_{Oi} + d_{iD} \\ d''_i := d_{Oi} + d_{iO} \end{cases}$$

onde O é o vértice Garagem e D é o vértice Destino Final. Construir as árvores **Pi1** e **PiHp** de acordo com sua posição relativa a cada rótulo de serviço.

Critério 1: Ordenar **PiHp** descendente (Mais Distante);

Critério 2: Ordenar **PiHp** ascendente (Mais Próximo).

Passo 3 (Estimar Número de Viagens): Estimar o número de viagens dos veículos (Frota Homogênea). - Critérios de [Negreiros & Maculan 92a], ou um limite *binpacking*, ou simplesmente fazer o número de viagens igual a infinito.

Passo 4 (Definir a Partida e Nova Viagem): Fazer marcador de partida PTOBS:= Origem (vértice). Abrir uma nova Rota enquanto ainda houver vértices a servir ou o número de viagens não se esgotar. Caso contrário terminar.

Passo 5 (Explorando e Aproveitando um Caminho): Pegar o primeiro rótulo em **PiHp** e atribuí-lo ao rótulo NoAt. Proceder o algoritmo de caminho mais curto ou mais rápido entre PTOBS e NoAt. Aplicar o procedimento APROVEITA para identificar todos os rótulos que estão entre PTOBS e NoAt, e os atribuir à Rota caso atendam às restrições do problema. À medida que os rótulos forem inseridos na Rota, eliminá-los de **Pi1**. Se ao atingir NoAt a sua inclusão viola as restrições, ir para o Passo 4. Caso contrário fazer PTOBS:= Pred(NoAt) do caminho gerado, retirar NoAt de **Pi1**, e colocar NoAt na rota. Ir ao passo 6.

Passo 6 (Novo Ponto de Observação): Identificar qual o tipo de rótulo de NoAt. Se NoAt é um rótulo de uma ligação fazer NoAt:=Suces(NoAt) tal que Suces(NoAt) $\neq$ PTOBS.

Se a lista de adjacências de NoAt $\in$ **Pi1** está vazia, fazer PTOBS:=NoAt, ir ao passo 5. Caso contrário, tomar o vértice *mais próximo* sobre o critério 1, ou o *mais distante* sobre o critério 2 desta lista de adjacências e que não viola as restrições do problema. Se tal vértice não existir, ir ao passo 4. Caso contrário fazer PTOBS:=NoAt, retirar NoAt de **Pi1**. Ir ao passo 6.

Na abertura de uma nova viagem (rota), no caso específico da coleta de lixo, a primeira pode iniciar e terminar em locais distintos como mostra o modelo do PCCLD. Neste caso, deve-se indicar na rota que o final do percurso foi atingido com um vértice diferente daquele de origem.

Uma vez que o algoritmo considera a movimentação em função da proximidade entre os diversos rótulos da rede, o procedimento pode ir tornando-se

atravessados a solução poderá ser melhorada significafivamente, porém é possível que piore em certos casos extremos.

O algoritmo MARCOLIX, conforme foi acima apresentado possui complexidade computacional $O(n^2 \ log \ k)$, onde n é o número de vértices da rede G', formada após a quebra de G e k é o número de vértices a servir, [Negreiros 90].

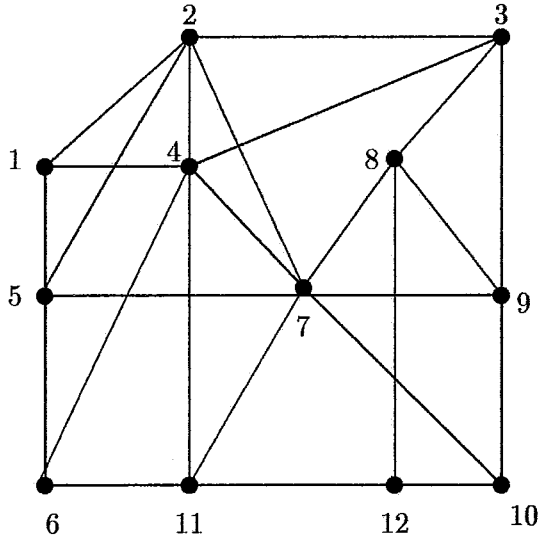
A formação das rotas pode variar segundo os critérios que foram adotados nos passos 2 e 6, com relação à forma de se explorar os rótulos de **PiHp**. Algumas das estratégias usadas nas avaliações do algoritmo MARCOLIX são aquelas mostradas na tabela VII.1.

O resultado da aplicação de cada estratégia, depende fortemente da rede. Os critérios apresentados não levam em conta os aspectos de preenchimento dos veículos, porém é factível a realização de percursos que consideram aqueles critérios propostos na heurística *Path Scanning*, [Pearn 91].

Como um exemplo da aplicação do algoritmo MARCOLIX, considere-se o problema exemplo proposto por [Pearn 91], que toma uma rede onde a desigualdade triangular não se verifica completamente no grafo. Sendo $Q = 10$ a capacidade do veículo. O algoritmo MARCOLIX evoluirá como segue:

Tabela VII.1: Descrição das estratégias usadas no algoritmo MARCOLIX

Estratégia	Descrição
VDD_d	**Passo 2:** Rota inicial e demais **PiHp** ordenadas pelo *rótulo mais distante.* **Passo 6:** tomar o rótulo adjacente `mais distante` de **Pi1**
VDD_p	**Passo 2:** Rota inicial e demais **PiHp** ordenados pelo *rótulo mais distante.* **Passo 6:** tomar o rótulo adjacente `mais próximo` de **Pi1**
VPP_d	**Passo 2:** Rota inicial e demais **PiHp** ordenados pelo *rótulo mais próximo.* **Passo 6:** tomar o rótulo adjacente `mais distante` de **Pi1**
VPP_p	**Passo 2:** Rota inicial e demais **PiHp** ordenados pelo *rótulo mais próximo.* **Passo 6:** tomar o rótulo adjacente `mais próximo` de **Pi1**
VDP_d	**Passo 2:** Rota inicial **PiHp** ordenados pelo *rótulo mais distante* e demais **PiHp** ordenados pelo *rótulo mais próximo.* **Passo 6:** tomar o rótulo adjacente `mais distante` de **Pi1**
VDP_p	**Passo 2:** Rota inicial **PiHp** ordenados pelo *rótulo mais distante* e demais **PiHp** ordenados pelo *rótulo mais próximo.* **Passo 6:** tomar o rótulo adjacente `mais próximo` de **Pi1**
VPD_d	**Passo 2:** Rota inicial **PiHp** ordenados pelo *rótulo mais próximo* e demais **PiHp** ordenados pelo *rótulo mais distante.* **Passo 6:** tomar o rótulo adjacente `mais distante` de **Pi1**
VPD_p	**Passo 2:** Rota inicial **PiHp** ordenados pelo *rótulo mais próximo* e demais **PiHp** ordenados pelo *rótulo mais distante.* **Passo 6:** tomar o rótulo adjacente `mais próximo` de **Pi1**



Lista de Elos		Lista de Elos	
Elo ij	($/$q_{ij}$)	7,8	(9/1)
1,2	(6/1)	7,9	(4/2)
1,4	(11/2)	7,10	(18/1)
1,5	(5/1)	7,11	(12/2)
2,3	(15/1)	8,9	(14/2)
2,4	(8/2)	4,6	(10/2)
2,5	(8/2)	8,12	(5/1)
2,7	(11/1)	10,9	(19/1)
3,4	(8/2)	10,12	(2/2)
3,8	(18/2)	11,12	(7/1)
3,9	(6/1)	6,11	(3/2)
4,6	(10/2)	7,8	(9/1)
4,7	(7/1)	7,9	(4/2)
4,11	(17/1)	7,10	(18/1)
5,6	(15/1)	7,11	(12/2)
5,7	(9/2)	8,9	(14/2)
6,11	(3/2)		

Figura VII.9: Problema exemplo proposto por Pearn 91.

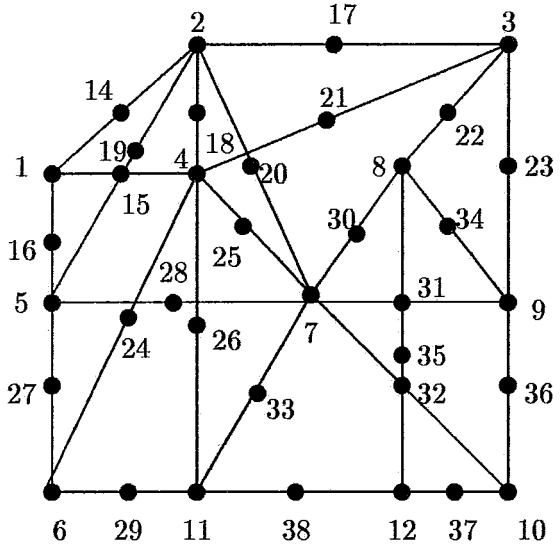


Figura VII.10: Quebra da rede - Transformação de arcos em nós.

Aplicando a estratégia VDD_d onde parte-se do vértice 1, tem-se:

PiHp

d"	56	56	55	51	51	50	43	42	41	39	37	32
Índice	22	37	36	22	38	34	29	23	32	26	30	24
q_{ij}	2	2	2	1	1	2	2	1	1	1	1	1

d"	32	30	30	29	27	25	23	19	18	12	11	6	5
Índice	31	18	21	25	17	27	20	28	19	33	15	14	16
q_{ij}	2	2	2	1	1	1	1	2	2	2	2	1	1

Rotas formadas com refinamento,

Rota 1: 1-(15)-4-(21)-3-(22)-8-(35)-12-(37)-10-(36)-9-1 ($D_1 = 81$, $Q_1 := 10$)
Rota 2: 1-(16)-5-(27)-6-(29)-11-(38)-12-8-(34)-9-(23)-3-(17)-2-1 ($D_2 = 76$, $Q_2 = 9$)
Rota 3: 1-5-(28)-7-(32)-10-8-(30)-7-(33)-11-(26)-4-(24)-6-4-1 ($D_2 = 103$, $Q_3 = 9$)
Rota 4: 1-5-(19)-2-(20)-7-(31)-9-7-(25)-4-(18)-2-(14)-1 ($D_2 = 66$, $Q_4 = 9$)

No exemplo acima, pode-se perceber que na formação das rotas há algum espaçamento entre demandas. Muitas etapas são realizadas com o veículo sem serviço, o que não é de fato bom. O Limite Inferior do problema acima obtido pelos procedimentos de

limite por *matching* propostos por [Golden & Wong 81] é 275, enquanto a solução obtida é de 326, afastando-se pois da solução em 18,5%.

Trocas locais entre rotas podem ser feitas como um procedimento de pós-otimização, a fim de atingir um resultado mais interessante, reduzindo as constantes quebras de serviço entre ligações.

Na coleta de lixo domiciliar, existe na maioria das vezes uma distribuição de carga bastante uniforme ao longo das regiões de periferia. Nas regiões mais verticais o mesmo não acontece, pois a constância da alternância entre grandes demandas e baixas demandas é um fator sempre relevante e que extrapola a relação combinatória da fase de empacotamento das cargas. Isto reflete fortemente na tomada de decisão do algoritmo MARCOLIX que pode ser ajustado de acordo, a tentar perceber estas constantes anomalias na instância em consideração.

VII.4 Transformação Linha

Um recurso da Teoria dos Grafos é a Transformação Linha, [Knuth 93]. Grosseiramente, a Transformação Linha é uma forma de representação em termos de matriz de adjacências de elos e arcos de um grafo não valorado. A literatura, como foi visto até aqui, não parece ter explorado esta técnica no tratamento de problemas de percurso ou no roteamento de veículos.

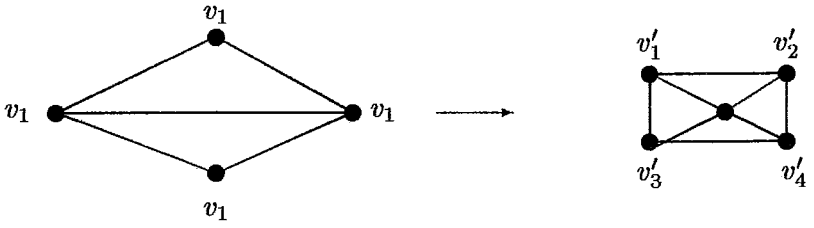
Formalmente, seja $G = (V, L)$, onde $L = (E, A)$, sem *loops*. Denomina-se transformação linha de G, o grafo Linha de G, o novo grafo $\mathcal{L}(V', L')$, $L' = (E', A')$, sendo $G \mapsto \mathcal{L}$ uma transformação onde os vértices de $\mathcal{L}$ indicam os elos/arcos de G, e os conjuntos E' e A' são formados através dos *caminhos adjacentes* que unem os dois elos/arcos adjacentes.

Alguns exemplos podem ser vistos na figura VII.11, onde se tem casos de transformação de grafos simples não direcionados e mistos. Pode-se então concluir que a lei de formação dos *caminhos adjacentes* considera os casos de um grafo geral, conforme figura VII.12.

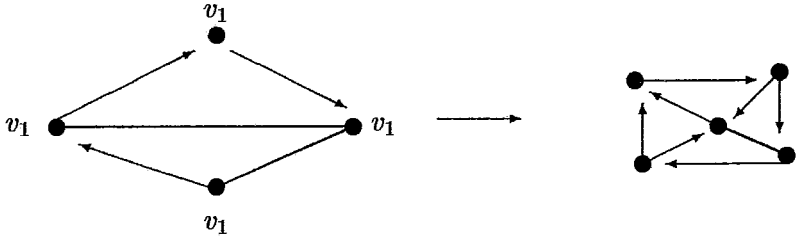
VII.4.1 Definições e Teoremas Básicos para Transformação Linha

Colocando mais formalmente os conceitos tem-se,

Definição VII.0 : (Formação do Grafo Linha) O Grafo Linha $\mathcal{L}$ de $G = (V, L)$ é formado do seguinte modo, [Knuth 93]:



(a) Transformação em um grafo Não Direcionado.



(b) Transformação em um grafo Misto.

Figura VII.11: Exemplos de Transformação Linha.

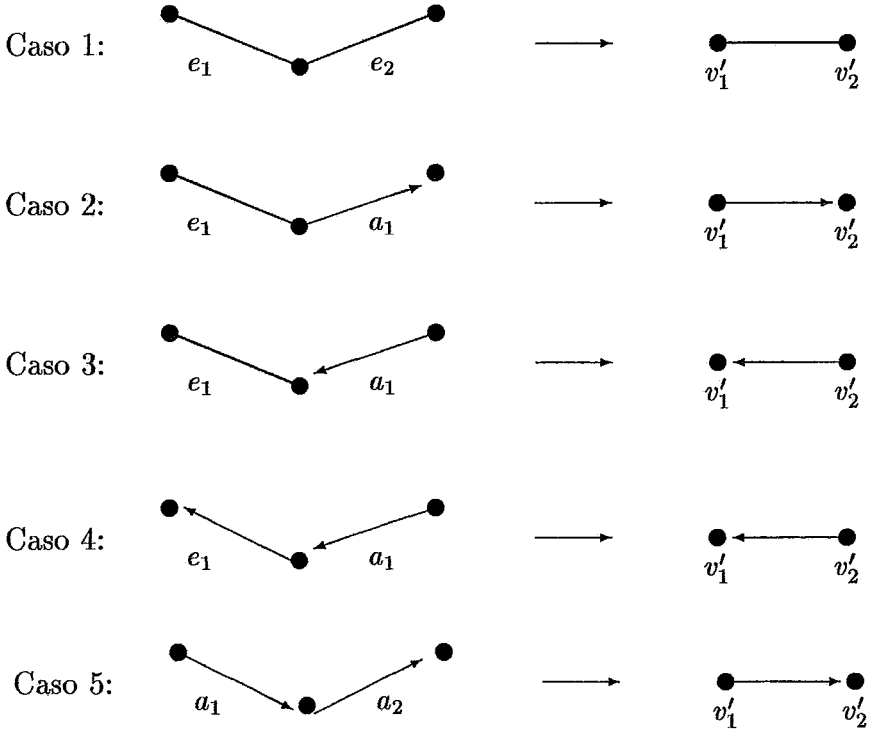


Figura VII.12: Casos de *Caminhos Adjacentes* na formação de um Grafo Linha.

1. Sendo L formado apenas por elos, o grafo linha de G tem um vértice para cada elo de G, e dois vértices em $(\mathcal{L})$ são adjacentes se e só se os elos correspondentes têm vértice comum em G.

2. Sendo L formado por elos e arcos, o grafo linha de G tem um vértice para cada arco/elo de G; há um arco do vértice u para o vértice v de $(\mathcal{L})$ se e somente se os arcos correspondentes a u terminam no vértice que inicia no arco correspondente a v.

Definição VII.1 : (Caminho Adjacente) É um caminho direto que se forma entre duas ligações com um único vértice comum, passando apenas por seus vértices.

Definição VII.2 : (Ligações de $\mathcal{L}(G)$) O número de ligações de $\mathcal{L}(G)$, supondo G não direcionado, $A = \emptyset$, depende essencialmente dos graus dos vértices de G, ou seja,

$$|E'| = -|E| + \frac{1}{2}\sum_{v_i} \rho(v_i),$$

onde $\rho(v_i)$ é o grau de v_i, $\forall\ v_i \in V$, [Guedes 90].

Definição VII.3 : (Digrafo Linha) Um Digrafo Linha é um grafo totalmente direcionado de $G_d = (V, A)$, onde para todo vértice de $\mathcal{L}_{d(G)}$ define-se uma relação direta $Pred(v) \rightarrow v \rightarrow Suces(v)$, $v \in V$, exemplo na figura VII.13.

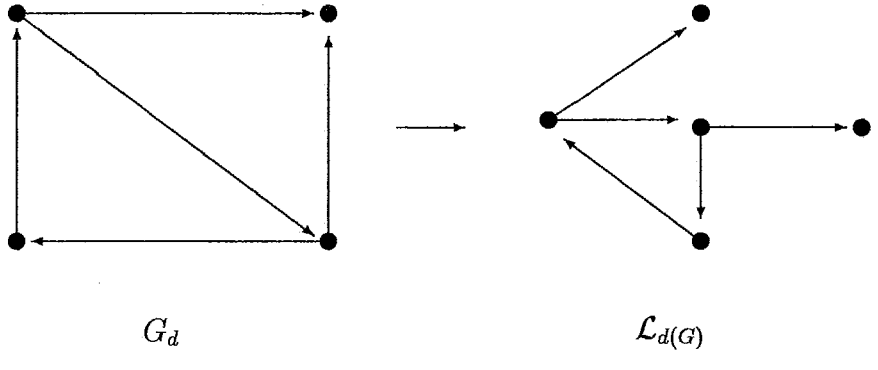


Figura VII.13: Uma transformação de G_d para um digrafo linha $\mathcal{L}_d$.

Teorema VII.1 : (f-Conectividade do Grafo Linha) Seja $G = (V, E, A)$ um grafo fortemente conexo, ou f-conexo, então sua transformação linha é f-conexa.

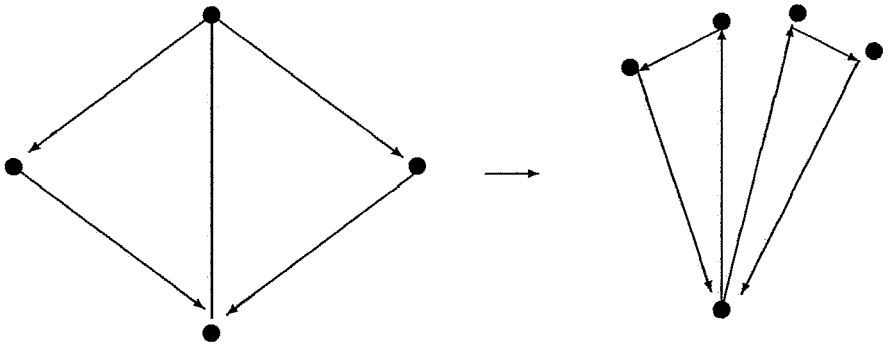
Prova

Se G é f-conexo, então todos os vértices de G são atingíveis a partir de qualquer vértice de G, logo existe sempre um caminho entre qualquer par de vértices de G. Se há um caminho entre todo par de vértices de G, então há sempre

um caminho adjacente *direcionado* entre pares de elos ou arestas adjacentes de G. Por este fato, haverá sempre um caminho entre quaisquer pares de vértices de $\mathcal{L}$, logo $\mathcal{L}$ é f-conexo. □

A figura VII.14 mostra a importância do teorema VII.1, uma vez que ele garantirá sempre a formação de percurso entre quaisquer pares de vértices de G tomando-se um $\mathcal{L}$ f-conexo como referência.

Importa agora conhecer a partir de um dado grafo direcionado ou misto G, o tempo necessário para formar sua transformação linha, $\mathcal{L}(G)$.



G f-conexo $\Rightarrow$ $\mathcal{L}$ f-conexo.

Figura VII.14: Formação linha f-conexa de um grafo f-conexo.

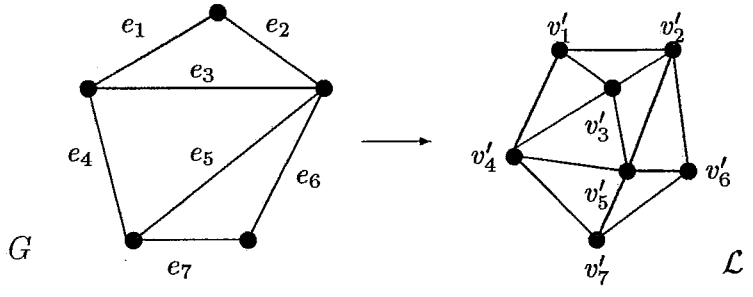
É fácil ver que para formar um Grafo Linha $\mathcal{L}$ a partir de $G = (V, E, A)$ f-conexo, são necessárias $O(n^4)$ operações, onde n é o número de vértices de G. Este valor, porém, é muito alto quando se tem redes com poucas ligações por vértice. Sabendo que o grau dos vértices de G é importante para conhecer o número de operações da transformação linha, certamente se pode estender este resultado para o problema onde a densidade do grafo é menor.

Supondo então $G = (V, L)$ um grafo f-conexo, onde $d(v_i)$ é o grau do vértice $v_i \in V$. Então o tempo necessário para a construção de $\mathcal{L}(G)$ será dado por $O(d^2_{\max} |V|)$, onde $d_{\max} = max\{d(v_i)| \ v_i \in V\}$.

Com o conhecimento de como formar um grafo $\mathcal{L}$ a partir de um grafo G, é necessário também considerar a correspondência entre operações elementares de grafos (caminhos, ciclos, etc.) em $\mathcal{L}$ e seus reflexos diretos em G. As definições a seguir tratam apenas daquelas operações que interessam à formação de rotas eulerianas em G.

<u>Definição VII.4 :</u> (Percurso Equivalente) Supondo que se deseja gerar

um caminho entre dois vértices $v'_i, v'_j \in \mathcal{L}, \forall\, ij$. O percurso equivalente $\mathcal{P}(v'_i, v'_j)$ corresponde ao conjunto de elos/arcos tomados entre v'_i a v'_j no grafo G, figura VII.15.



$$\mathcal{P}(v'_1, v'_5) = v'_1\ v'_4\ v'_5\ = (e_1 e_4)(e_4 e_5) \equiv\ e_1\ e_4\ e_5$$

Figura VII.15: Composição de um percurso equivalente.

Pode-se notar da figura VII.15 que todos os elementos que estão no meio do percurso entre v'_i e v'_j são repetidos duas vezes, com cortes curtos elimina-se os elementos repetidos e tem-se o percurso em G.

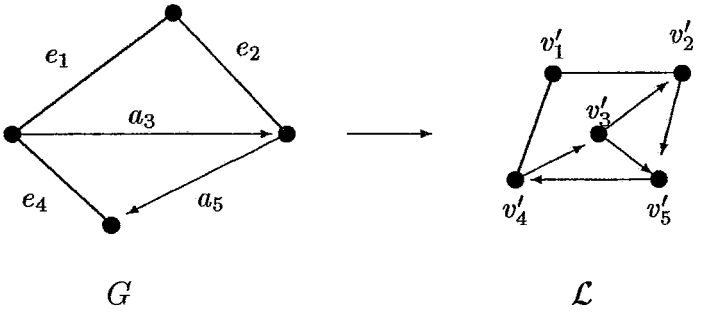
Assumindo-se que os vértices origem e destino são valorados de uma unidade, e que seu peso será acrescido no caminho, o percurso equivalente $\mathcal{P}(v'_i, v'_j)$ será dado pela soma dos elos/arcos no caminho de v'_i a v'_j e a soma dos pesos de v'_i e v'_j dividido por dois. Neste caso, nota-se que os pesos incidem apenas no início e no fim do cálculo do percurso, correspondendo também ao valor da cardinalidade do percurso gerado pela cadeia formada em G.

Definição VII.5 : (Circuito em $\mathcal{L}$ - Circuito Equivalente $\mathcal{C}$) Um circuito em $\mathcal{L}$ é formado por um caminho fechado que parte de um vértice origem, percorre um subconjunto de vértices de $\mathcal{L}$ e retorna ao vértice origem. Neste caso, o percurso formado pelo circuito de $\mathcal{L}$ tem um equivalente em G porém com todas as ligações dobradas, figura VII.16.

Definição VII.6 : (Caminho-Circuito $P_{\mathcal{L}}C_G$) Um caminho-circuito é definido quando se tem em $\mathcal{L}$ um caminho entre dois vértices, cujo caminho equivalente em G é um circuito. Neste caso, as únicas ligações repetidas são aquelas referentes aos vértices internos do caminho em $\mathcal{L}$, figura VII.17.

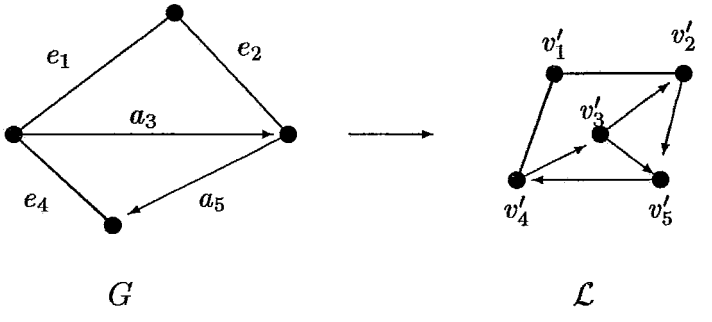
Definição VII.7 : (Circuito Hamiltoniano Linha $\mathcal{H}$ - Circuito Euleriano) É um circuito formado em $\mathcal{L}$ que percorre todos os seus vértices apenas uma vez. Assim, como no circuito $\mathcal{C}$, todas as ligações aparecem dobradas no circuito equivalente em G. Porém com cortes curtos, pode-se eliminar os elementos replicados em seguida incluir (replicar) aqueles elos que estão desconectando o percurso de modo a torná-lo euleriano, figura VII.18.

Na figura VII.19, vê-se a composição de um circuito euleriano a partir



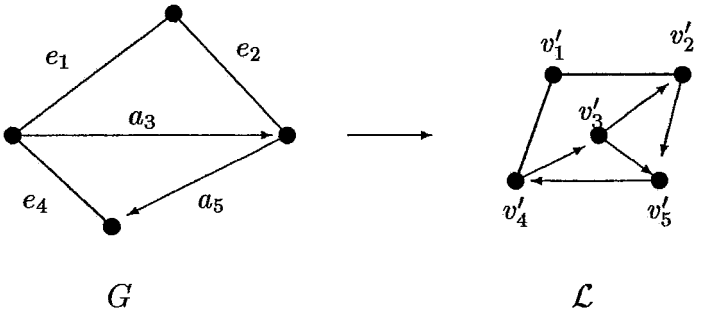
$$\mathcal{C}(v_2', v_5', v_4', v_3', v_2') = (e_2a_5)(a_5e_4)(e_4a_3)(a_3a_2) \equiv e_2a_5e_4a_3$$

Figura VII.16: Composição de um circuito equivalente.



$$\mathcal{P}(v_2', v_5', v_4', v_3') = (e_2a_5)(a_5e_4)(e_4a_3) \equiv e_2a_5e_4a_3$$

Figura VII.17: Composição de um caminho-circuito.



$$\mathcal{P}(v_2', v_5', v_4', v_3') = (e_2a_5)(a_5e_4)(e_4a_3) \equiv e_2a_5e_4a_3$$

Figura VII.18: Exemplo de um circuito hamiltoniano linha e circuito euleriano após composição.

de uma transformação linha em um grafo não orientado. Nota-se que a etapa de inserção de novas arestas de modo a manter conexo o grafo, depende fortemente de o grafo gozar da propriedade de custo triangular. Neste aspecto, a decisão na remontagem do circuito pode depender fortemente da instância em análise.

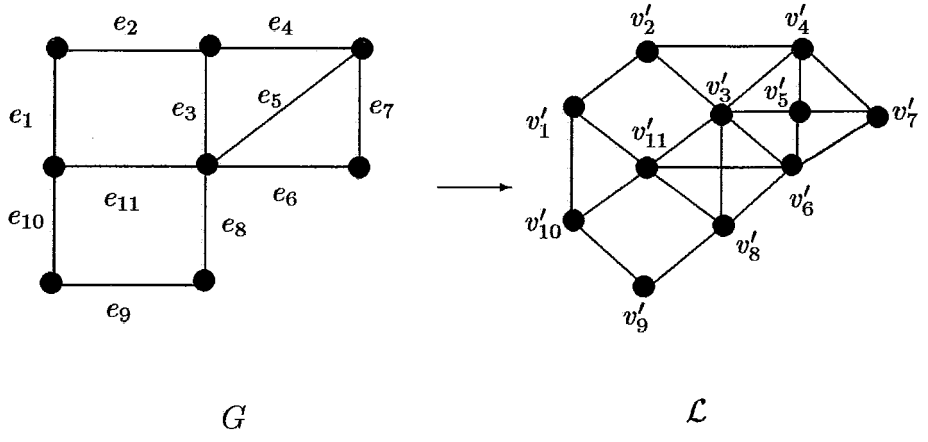


Figura VII.19: Grafo linha de um grafo não orientado.

$$\begin{aligned} &\mathcal{C}(v'_1, v'_2, v'_3, v'_4, v'_5, v'_7, v'_6, v'_{11}, v'_8, v'_9, v'_{10}, v'_1) \\ = &(e_1e_2)(e_2e_3)(e_3e_4)(e_4e_5)(e_5e_7)(e_7e_6)(e_6e_{11})(e_{11}e_8)(e_8e_9)(e_9e_{10})(e_{10}e_1) \\ &\equiv e_1e_2e_3e_4e_5e_7e_6e_{11}e_8e_9e_{10}e_1 \end{aligned} \tag{VII.17}$$

Aumentando VII.17, tem-se: $C_E = e_1e_2e_3(e_3)e_4e_5(e_5)e_7e_6e_{11}(e_{11})e_8e_9e_{10}e_1$. Onde os elos em (.) são repetidos.

VII.4.2 Matriz de Adjacências Valorada de $\mathcal{L}$

A princípio, as regras de formação dos custos das arestas podem ser extraídas diretamente da formação das ligações de $\mathcal{L}$, a partir da matriz de pesos de G, aplicando-se a seguir os limites considerados. Porém, nos casos onde se avaliam percursos e circuitos onde os vértices não são adjacentes, alguns problemas podem ocorrer na formação da equivalência. Para evitar isto, sugere-se alguns critérios para redimir os casos problemáticos, como seguem.

<u>Caso 1:</u> Circuito formado por vértices de $\mathcal{L}$ onde v'_i é adjacente a v'_{i+1}.

<u>Caso 2:</u> Circuito formado por vértices de $\mathcal{L}$ onde existe pelo menos um v'_i que não é adjacente a v'_{i+1}, $v'_i \in V'$.

(1) Medida para o Caso 1

O peso dado ao elo/arco $v'_i v'_j$ será relativo a: $c_{ij}^{v'_i}$ ou a $c_{jk}^{v'_j}$.

1. Se $c_{ij}^{v'_i}$ for adotado, o percurso entre quaisquer pares $v'_s, v'_t \in V'$ terá um custo igual ao custo do caminho mais curto de v'_s a v'_t acrescido do valor do elo/arco destino.
2. Se $c_{ij}^{v'_j}$ for adotado, o percurso entre quaisquer pares $v'_s, v'_t \in V'$ terá um custo igual ao custo do caminho mais curto de v'_s a v'_t acrescido do valor do elo/arco origem.

(2) Medida para o Caso 2

O peso dado ao elo/arco $v'_i v'_j$ será relativo a:

1. $c_{ij}^{v'_i} + c_{jk}^{v'_j}$. Assim, após o cálculo do percurso entre quaisquer pares de vértices $v'_s, v'_t \in V'$ o custo total deverá ser dividido por dois.

Ambos os cálculos podem ser aplicados para a avaliação dos custos de percurso entre quaisquer dois vértices de $\mathcal{L}$. Porém, quando um circuito é acompanhado de um problema de percurso com o objetivo de resolver por exemplo um PCR, alguns cuidados devem ser tomados na escolha da medida (1) ou (2). Nos experimentos que fiz, os melhores resultados foram atingidos pela aplicação da medida (1).1.

Nota-se que a solução do problema de percursos de veículos em arcos não é direta como alguém poderia inferir. Algumas aplicações porém não têm esta necessidade, isto será discutido na seção VII.5

VII.4.3 Aplicações da Transformação Linha

Uma vez que se vem discutindo o uso das trasformações no PGR ou PRAD, a transformação linha se aplica a este caso, desde que os vértices demanda sejam expandidos a elos com demanda, conforme proposta de [Golden & Wong 81].

Um algoritmo apropriado para a aplicação da transformação linha considerando o problema do Carteiro Chinês, ou do Carteiro Rural para os quais quaisquer instâncias relacionadas, pode ser definido como segue.

Algoritmo Geral PRAD-TL - para o PCC/PCR.

Entrada: Grafo G, origem e ligações por onde o Carteiro vai passar.

Saída: Custo e descrição da Rota do Carteiro.

Passo 1: Expandir o vértice origem para um elo de custo zero, e rotular o novo vértice incluindo-o em V como um vértice origem.

Passo 2: Aplicar a Transformação Linha de G e identificar na rede $\mathcal{L}$ as ligações de G marcadas para atendimento.

Passo 3: Gerar uma matriz de percursos entre os elementos identificados em $\mathcal{L}$.

Passo 4: Aplicar um algoritmo do PCV sobre os vértices identificados em $\mathcal{L}$.

Passo 5: Gerar o percurso-circuito $\mathcal{P} - C_E$ da solução obtida a partir do passo 4. Aplicar cortes curtos. Terminar.

O algoritmo acima, pode considerar qualquer problema de percurso do Carteiro, dentre os vários aqui expostos no capítulo VI. Apesar de não se garantir que a transformação favorecerá a uma solução ótima, mesmo que se use algoritmos exatos para o PCV. Isto se explica devido ao fato de que a solução obtida dependerá fortemente da solução do ciclo hamiltoniano encontrada em $\mathcal{L}$. Se um ciclo hamiltoniano for encontrado em $\mathcal{L}$ a solução retornará em seu custo o perímetro de G em sua primeira instância, ou seja, antes de ser aumentada. Ao se aumentar a solução com elos/arcos de G de modo a permitir a continuidade da solução, esta será ϵ-aproximativa.

Em vários testes realizados sobre grafos conhecidos da literatura, atingiu-se expressivos resultados usando tanto o método exato B&B quanto os métodos heurísticos, ver tabela VII.2.

Uma extensão do problema *Stacker Crane* pode ser obtida desta técnica, onde G é um grafo misto e deseja-se atender certos elos/arcos não adjacentes. A formulação proposta mantém-se robusta para este caso.

Quanto ao problema do Carteiro Íngreme, a restrição de custo existente conforme a travessia do elo deverá ser considerada no momento da formação do custo das ligações, requerendo neste caso uma pequena variação na formulação das ligações de $\mathcal{L}$, figura VII.20.

Para todos os elos onde esta condição mostra-se considerável, a transformação manter-se-á robusta. Ou seja, poderá ser extendido para qualquer caso do PGR ou PRV com ou sem restrições adicionais. Para que isto ocorra, basta que no passo 4 do algoritmo geral aplique-se o algoritmo apropriado ao problema. A transformação manter-se-á robusta também nestas situações.

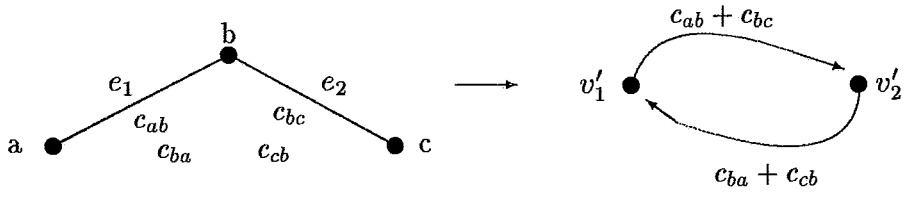


Figura VII.20: Mudança proposta à TL para o problema do Carteiro Íngreme.

Uma outra grande vantagem desta técnica é quanto à aplicação de restrições de tráfego impostas pelas leis de trânsito (retorno proibido, dobrar à direita ou à esquerda), dentre outras. Outros usos importantes da técnica são: Na formação de regiões de atendimento de hospitais, postos de saúde, setores censitários, regiões e sub-regiões de coleta de lixo domiciliar, zonas eleitorais, etc.

VII.4.4 Alguns Resultados Experimentais

Os resultados que são reportados na tabela VII.2, consideram uma gama de problemas da literatura e alguns novos problemas que são introduzidos com o objetivo de colocar um posicionamento mais vasto no teste de instâncias teóricas e práticas obtidas de minha experiência em campo com a aplicação de algoritmos de percursos em arcos.

De forma mais específica, o *Problema* é um grafo cujos atributos gerais de custo de vértices e ligações mantêm-se os mesmos para diversas formulações matemáticas que o utilizam.

Os algoritmos TL e MARCOLIX, foram implementados no compilador Turbo Pascal V7.0 –, Borland International, em ambiente MS-DOS num microcomputador PC-IBM/PENTIUM 100MHz, utilizando os recursos gráficos do sistema **SisGRAFO**.

O problema 1 aborda o grafo exemplo proposto por [Larson & Odoni 81], para o PCC não orientado, uma instância para o PRAD também é considerada. No problema 2, tem-se o mesmo grafo do problema 1, porém agora o foco é o problema do Carteiro Rural não direcionado, quatro componentes conexas formam a instância.

O problema 3 considera o grafo proposto por [Christofides 73], para os casos do PCC não direcionado, e para o PRAD. No problema 4, tem-se o grafo proposto por [Ulosoy 85], onde se considera o PCC e o PRAD não direcionados. No problema 5, tem-se o grafo proposto por [Burkard & Derigs 80], onde se extende a análise ao PCC, e duas instâncias do PRAD não orientados.

Nos problemas 6 e 7, tem-se os grafos exemplos propostos por [Pearn 89] e por [Pearn 91] respectivamente para o PRAD. Estes exemplos foram cuidadosamente preparados pelo autor com o objetivo de testar limites inferiores e o desempenho de heurísticas específicas relacionadas ao PRAD.

O problema 8 aborda o grafo exemplo proposto por [Christofides et al 86], para o PCR direcionado, considerando três componentes conexas.

O problema 9 considera um grafo misto, onde se avalia o desempenho das transformações considerando o PCC e uma instância do PRAD. No problema 10 tem-se o mesmo grafo do problema 9, porém considerando uma aplicação do PCR misto sobre duas componentes conexas.

O problema 11 considera um grafo misto razoavelmente grande onde se observa uma variação do Problema *Stacker Crane*, no qual deseja-se fazer o percurso do *Crane* (guindaste) sobre um conjunto selecionado de elos/arcos da rede.

Os problemas 12 e 13 consideram dois grafos mistos de duas sub-regiões de coleta de lixo domiciliar da cidade de Fortaleza, Aldeota (474 e 476), cujos dados foram extraídos de um trabalho de campo, conforme [Negreiros 90].

O problema 14 também considera um grafo misto extraído da aplicação teste de [Negreiros et al 92a] para uma sub-região (BA-237) de coleta de lixo domiciliar e especial, no bairro da Urca - Rio de Janeiro.

Tabela VII.2: Comparação entre as Transformações - PCC.

Problema	Tipo	Grafo	$\|N\|$	$\|L\|$	$\|S\|$	LI	TL	MARCOLIX
1	PCC	Não Direcionado	13	19	19	3830	4090	3830
							0.99s	1.70s
3	PCC	Não Direcionado	13	22	22	294	355	307
							1.43s	2.41s
4	PCC	Não Direcionado	11	18	18	720	786	746
							0.88s	2.54s
5	PCC	Não Direcionado	51	78	78	6700	8941	7066
							48.72s	165.0s

Problema	Tipo	Grafo	$\|N\|$	$\|L\|$	$\|S\|$	LI	TL	MARCOLIX
2	PCR	Não Direcionado	14	19	8	2560	2560	3230
							0.44s	1.10s
8	PCR	Direcionado	23	46	21	150	150	150
							1.49s	1.37s
9	PCR	Misto	14	16	16	219	219	267
							1.15s	1.37s
10	PCR	Misto	14	16	6	129	129	199
							0.27s	2.58s
11	PSCm	Misto	165	242	15	347	347	347
							7.36s	18.84s

Problema	Tipo	Grafo	$\|N\|$	$\|L\|$	$\|S\|$	LI	TL	MARCOLIX
1	PRAD	Não Direcionado	14	19	19	-	(3) 5360	(3) 4900
							3.85s	1.10s
3	PRAD	Não Direcionado	13	22	22	(5) 310	(5) 377	(5) 356
							8.57s	2.63s
4	PRAD	Não Direcionado	11	18	18	-	(6) 1121	(7) 1160
							3.79s	2.08s
4	PRAD	Não Direcionado	11	18	18	-	(5) 1023	(6) 1021
							5.05s	1.87s
5	PRAD	Não Direcionado	51	78	78	-	(7) 28481	(7) 28384
							48.81s	29.0s
5	PRAD	Não Direcionado	51	78	78	-	(4) 17649	(4) 17379
							53.22s	28.94s
6	PRAD	Não Direcionado	13	25	25	(4) 275	(4) 359	(4) 314
							7.14s	2.58s
7	PRAD	Não Direcionado	12	23	23	(7) 474	(7) 538	(7) 587
							2.48s	2.53s
9	PRAD	Misto	14	16	15	-	(4) 361	(5) 443
							2.31s	1.37s
12	PRAD	Misto	91	154	130	-	(4) 234154	(2) 126205
							114.96s	18.84s
13	PRAD	Misto	94	163	124	-	(4) 154730	(3) 112830
							198.11s	30.76s
14	PRAD	Misto	69	99	52	-	(2) 24222	(2) 30427
							19.60s	20.16s

$|N|$ - é o número de vértices do grafo;

$|L|$ - é o número de ligações do grafo (elos e arcos);

$|S|$ - é o número de ligações de serviço;

LI - é o limite inferior do problema considerado pela literatura;

TL - resultado usando Transformação Linha;

MARCOLIX - resultado usando o algoritmo MARCOLIX.

As soluções acima colocadas são relativas aos melhores resultados obtidos com as aplicações das heurísticas após a realização das transformações. O tempo (s) reflete o processo de execução do(s) algoritmo(s) sem levar em conta o tempo levado para a realização das transformações.

No caso da transformação linha, foi utilizado o método B&B proposto por [Syslo et al 83] para os casos em que $|S| \leq 40$, e o método Or-Otimal, [Or 76], nos casos onde $|S| > 40$, relativos aos problemas PCC, PCR e PSCm, e como algoritmos de refinamento nas fases finais dos procedimentos de roteamento em nós. O algoritmo utilizado para a análise dos problemas PRAD foi o método de Clarke & Write modificado, proposto por [Negreiros 93], considerando as versões paralelas e sequencial.

Para o procedimento MARCOLIX, todos os resultados apresentados foram refinados com o procedimento 2-Otimal, considerando a abordagem simétrica e assimétrica do PCV. Um detalhe importante, já comentado na seção VII.3, é que em algumas estratégias a aplicação do método otimal piora consideravelmente a solução inicial obtida uma vez que após sua aplicação tem-se que garantir a travessia completa dos elos. Uma mudança de estratégia, nestes casos, pode ajudar a sair deste problema, sendo possivel que neste caso haja uma possibilidade de redução no custo da solução inicial de forma expressiva.

O algoritmo MARCOLIX mostrou-se eficaz em muitas instâncias do PCC não direcionado tendo atingido uma distância máxima da solução ótima de 5,5% em uma instância com um número razoável de vértices e ligações (Problema 5 - PCC). Também atingiu o ótimo em instâncias curtas. Já os algoritmos usando TL não tiveram o mesmo sucesso para este problema, como se previa.

Nas instâncias do PCR, a TL mostrou-se mais efetiva, tendo atingido a solução ótima em todas as instâncias analisadas. Já o algoritmo MARCOLIX mostrou-se menos efetivo, tendo se afatado da solução ótima de forma incontrolável, apesar de em algumas instâncias ter atingido a solução ótima.

Por fim nas instâncias relativas ao PRAD o algoritmo MARCOLIX mostrou-se melhor naquelas onde foram fornecidos pela literatura os limites inferiores do problema, porém os resultados obtidos ficaram distantes das soluções em até 23,8%. A TL se mostrou mais efetiva em uma das instâncias relativas a situação real,

tendo atingido uma melhoria em relação ao algoritmo MARCOLIX de 25,6%. Já o algoritmo MARCOLIX superou em uma das instâncias reais a TL em até 85,5%. Nos problemas não direcionados, porém, os métodos se mostraram muito próximos alternando entre os melhores resultados.

Apesar da TL ter sido menos eficiente em muitas instâncias, ela mostrou-se poderosa em todos os casos analisados do PCR tendo atingido a solução exata. Como é conhecido da literatura, o algoritmo de Clarke & Write não tem um comportamento estável quanto à proximidade do ótimo, isto de certa forma mostra o *por quê* da grande instabilidade na solução do PRAD. Mais ainda, as heurísticas otimais também não são sempre tão efetivas no PCV assimétrico como já se sabe do capítulo III.

Pelos resultados obtidos uma tarefa interessante de pesquisa seria o efeito da transformação linha no PRAD considerando algoritmos mais agrupativos como os de pétalas e *Sweep*, para instâncias onde as ligações com demanda formam uma única componente conexa. Já para os casos onde se tem várias componentes conexas, um trabalho de pesquisa interessante pode surgir daí.

Uma vez que algoritmos com comportamento 0.5-aproximativos para o problema são de fato um problema inicial para o tratamento de instâncias do PRAD não orientado, justifica-se o fato de se ter diferentes métodos de solução para este problema, com a intenção de tratar uma mesma instância na obtenção dos melhores resultados.

As instâncias tomadas, mostram de forma clara o desempenho das transformações sobre diversos tipos de grafos, mostrando-se robusta no tratamento dos problemas, porém necessitando de um trabalho adicional para torná-las mais efetivas, considerando a distância aos limites inferiores. No caso da transformação linha, por exemplo, métodos otimais de grande porte devem ser ativados na solução de problemas do PCC e PRAD, enquanto que uma fase de trocas locais entre rotas no algoritmo MARCOLIX poderia ter uma relevante contribuição.

VII.5 Minimizando o Esforço no Percurso em Ligações

Algumas vezes, nas aplicações de roteamento em arcos (ruas), é necessário que os percursos construidos sejam ponderados por certos critérios de continuidade. Elementos importantes tais como manter-se em curso, dobrar à direita ou à esquerda podem significar uma enorme perda de tempo num serviço doméstico diário. Com o objetivo de tratar este problema alguns algoritmos específicos foram construidos para este fim.

O algoritmo para encontrar um percurso euleriano, apresentado no capítulo VI, não estão adaptados para resolver o problema gerado por muitos retornos em U, ou muitas tomadas à esquerda, fato que causa ao trabalhador de rua a mudança de lados por muitas vezes, além disso, alterna frequentemente entre a execução do serviço de coleta/entrega e a passagem sem serviço. Como já foi colocado, uma boa rota é aquela que busca fazer longas passagens com serviço assim como longas sem serviço.

Um algoritmo foi inicialmente desenvolvido com o intuito de gerar esta configuração inicial da rota. A primeira parte do procedimento é examinar cada vértice da rede aumentada individualmente e fazer algo para casar os arcos que saem do vértice aos que entram, [Bodin & Krush 79].

Para juntar os pares predecessores-sucessores neste vértice, um problema de atribuição é resolvido. Os custos d_{ij} do problema de atribuição denotam a penalidade que estaria incorrida pela atribuição do i-ésimo arco predecessor com o j-ésimo arco sucessor. A penalidade d_{ij} é uma função do tipo de mudança que o motorista faria, se ele mudaria entre uma travessia de um arco com serviço e em seguida em outro sem serviço ou atravessando de um lado para outro. A solução deste problema retornaria o melhor emparelhamento global dos arcos predecessor sucessor.

Pode-se modelar este problema como um Problema de Atribuição com Gargalo, [Garfinkel 70]. O objetivo do problema de atribuição com gargalo é minimizar o mínimo (d_{ij}) enquanto restringe a colocação de atribuições viáveis. Isto daria uma solução cuja penalidade total deve ser maior do que um problema de atribuição simples mas que reduz tanto quanto possível a pior situação.

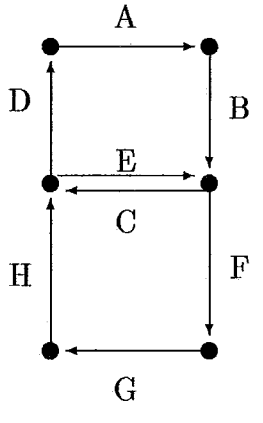


Figura VII.21: Avaliando subciclos.

Todo arco da rede aumentada tem um arco sucessor para um vértice e um arco predecessor para outro. Portanto, iniciando-se em algum vértice, a rota pode ser facilmente enumerada. Para cada componente na rede aumentada, o em-

parelhamento ótimo pode gerar mais do que um subciclo, desde que setando os emparelhamentos predecessor-sucessor nenhum cuidado seja tomado para permitir que a rota de Euler exista. Por exemplo, na figura VII.21 tem-se dois subciclos ABCD e EFGH (cada subciclo não contém retornos em U), enquanto que o ciclo ABCEFGHD contém todos os arcos, porém tem um retorno em U. No caso, o procedimento que se propõe encontraria os dois subciclos ABCD e EFGH.

Para ilustrar melhor o procedimento proposto por [Bodin & Krush 79], considera-se na figura VII.22 um grafo euleriano direcionado. Cada vértice no grafo será analisado como uma atribuição *pred-suces* de ramos incidentes a este vértice. Tomando esta atribuição as penalidades são dadas:

- Seguir em Frente: 0 ponto;
- Dobrar à Direita: 1 ponto;
- Dobrar à Esquerda: 4 pontos;
- Retorno em U: 8 pontos;
- Sem-serviço com serviço ou vice-versa: 5 pontos.

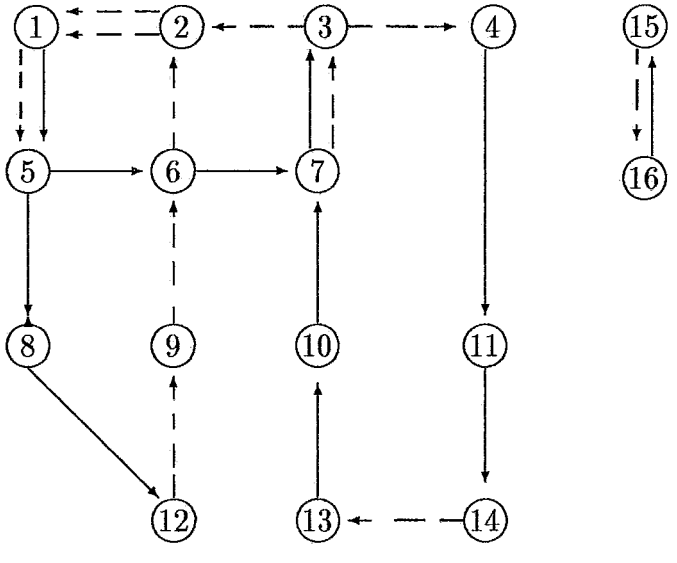


Figura VII.22: Grafo Euleriano para avaliação de rota com esforço mínimo.

Como exemplo, a matriz de penalidades para o vértice 3 seria,

A_i — A_s	$3 \rightarrow 2$	$3 \rightarrow 4$
$7 \rightarrow 3$	(5+4)	(5+1)
$7 \rightarrow 3$	(0+4)	(0+1)

Solução: $7 \rightarrow 3 \rightarrow 2$ e $7 \rightarrow 3 \rightarrow 4$.

Procedendo para todos os vértices tem-se na tabela VII.3 o quadro de serviço da rede.

Tabela VII.3: Quadro de atribuição de serviço.

Vértice Central		
1	$2 \rightarrow 1 \rightarrow 5$	$2 \rightarrow 1 \rightarrow 5$
2	$3 \rightarrow 2 \rightarrow 1$	$6 \rightarrow 2 \rightarrow 1$
3	$7 \rightarrow 3 \rightarrow 2$	$7 \rightarrow 3 \rightarrow 4$
4	$3 \rightarrow 4 \rightarrow 11$	
5	$1 \rightarrow 5 \rightarrow 8$	$1 \rightarrow 5 \rightarrow 6$
6	$5 \rightarrow 6 \rightarrow 7$	$9 \rightarrow 6 \rightarrow 2$
7	$6 \rightarrow 7 \rightarrow 3$	$10 \rightarrow 7 \rightarrow 3$
8	$5 \rightarrow 8 \rightarrow 12$	
9	$12 \rightarrow 9 \rightarrow 6$	
10	$13 \rightarrow 10 \rightarrow 7$	
11	$4 \rightarrow 11 \rightarrow 14$	
12	$8 \rightarrow 12 \rightarrow 9$	
13	$14 \rightarrow 13 \rightarrow 10$	
14	$11 \rightarrow 14 \rightarrow 13$	
15	$16 \rightarrow 15 \rightarrow 16$	
16	$15 \rightarrow 16 \rightarrow 15$	

Com o objetivo de formar os subciclos, a tabela VII.3 deve ser lida como segue. Iniciar com o vértice origem, (1). O sucessor do vértice 1 é o vértice 5. Ir para o vértice 5 marcando o primeiro conjunto *pred-nó-suces* de 1 como atendido (2-1-5). Dentre os sucessores de 5 tomar aquele que não que satisfaz 1-5-X. Tomar 1-5-8 e pegar uma nova combinação em 8 que satisfaça 5-8-X. Proceder até que o vértice origem seja atingido novamente. Tomar o vértice do primeiro conjunto *pred-nó-suces* da tabela não marcado como atendido, proceder como antes até que todos os elementos da tabela tenham sido marcados como visitados.

A solução do problema pode então ser vista na tabela VII.4.

O problema agora é de reunir os subciclos encontrados no passo anterior com um componente do grafo euleriano em uma rota sobre todos os seus arcos, checando se quaisquer dois subciclos contêm o mesmo número de nodos. Se tiverem, então eles podem ser ligados de modo a formar um subciclo maior. Este subciclo maior atravessa a rota no primeiro subciclo até que atinja o vértice comum. Ele então segue para completar o segundo subciclo iniciando deste vértice comum para ambos os subciclos e finalmente completar a rota sobre o primeiro subciclo.

Tabela VII.4: Quadro de atribuição de serviço.

Subciclo Aberto	Sub-ciclo
1	1-5-8-12-9-6-2-1-5-6-7-3-2-1
2	3-4-11-14-13-10-7-3
3	15-16-15

No exemplo que vem-se ilustrando, o vértice comum aos subciclos 1 e 2 é o vértice 7. Assim estes dois subciclos podem ser unidos pela conecção de predecessor do arco $< 6,7 >$ no subciclo 1 ao arco sucessor do arco $< 7,3 >$ no subciclo 2. Os subciclos resultantes são aqueles apresentados na figura VII.23.

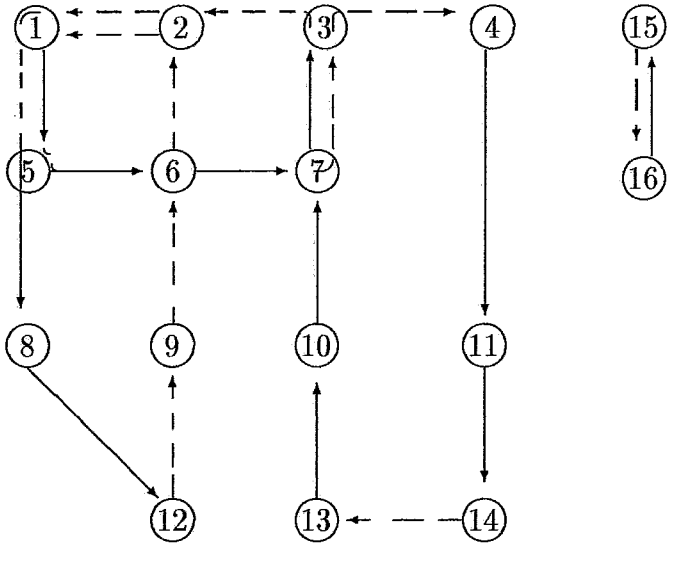


Figura VII.23: Juntando subciclos gerados.

Da solução obtida pode-se verificar facilmente que é comum para três ciclos passarem pela mesma interseção. Uma estratégia para a redução de subciclos foi proposta por [McBridge 82], como segue.

Algoritmo de McBridge para **Redução de Subciclos**

Passo 1: Se todo subciclo está combinado. Terminar.

Passo 2: Combinar ciclos nas interseções que têm três ciclos comuns próximos. Se um retorno em U for introduzido após tentar todas as possíveis formas de combinar os ciclos, pular a interseção. Se os ciclos estão combinados ir para o passo 1.

Passo 3: Combinar ciclos em interseções que têm dois ciclos comuns próximos. Se um retorno em U for introduzido, pular a interseção. Se os ciclos são combinados, ir para o passo 1.

Passo 4: Combinar ciclos nas interseções que têm três ciclos comuns próximos. Se os ciclos estão combinados ir para o passo 1.

Passo 5: Combinar ciclos que têm dois ciclos comuns por último. Ir ao passo 1.

[McBridge 82] usou a seguinte pontuação para a movimentação dos veículos:

- Seguir em Frente: 0 ponto;
- Dobrar à Direita: 1 ponto;
- Dobrar à Esquerda: 5 pontos;
- Retorno em U: 11 pontos;

Em cada passo no procedimento acima, cada interseção seria verificada. Se nenhuma interseção puder ser considerada onde os ciclos são combinados, então segue-se para o próximo passo. Se os ciclos estão combinados, retorna-se ao passo 1 e reinicia-se os passos novamente.

Há apenas três modos de se combinar subciclos passando por um vértice comum, conforme [McBridge 82]. Para isto, o autor usou a seguinte estratégia: Supondo-se que inicialmente os arcos a, b e c estão atribuidos aos arcos a', b' e c' como indicado pelos pares ordenados (a, a'), (b, b') e (c, c'), uma nova atribuição é tentada: (a, b'), (b, c') e (c, b'). Se nenhum retorno em U é introduzido ela é mantida, caso contrário, a atribuição (a, b'), (b, c') e (c, b') é considerada. No passo 5, esta última atribuição mantém-se par se um retorno em U é introduzido.

Os procedimentos de [Bodin & Krush 79] e [McBridge 82] são complementares, apesar de haver uma certa divergência quanto aos valores dos custos de movimentação. Uma vez que os autores utilizaram a abordagem *predecessor - nó - sucessor* pode-se inferir diretamente que a transformação linha poderá resolver diretamente o problema, desde que o método de solução empregado seja exato.

Mais formalmente, seja pois $G_E = (V, A)$ um grafo direcionado e euleriano que está definido pela rota euleriana sobre as ligações de G. Um grafo T_G é um grafo direcionado onde os vértices representam os arcos de G e os arcos representam os movimentos realizados entre dois arcos adjacentes de G. Nota-se, como foi dito, que T_G é uma transformação linha de G, porém os custos relativos

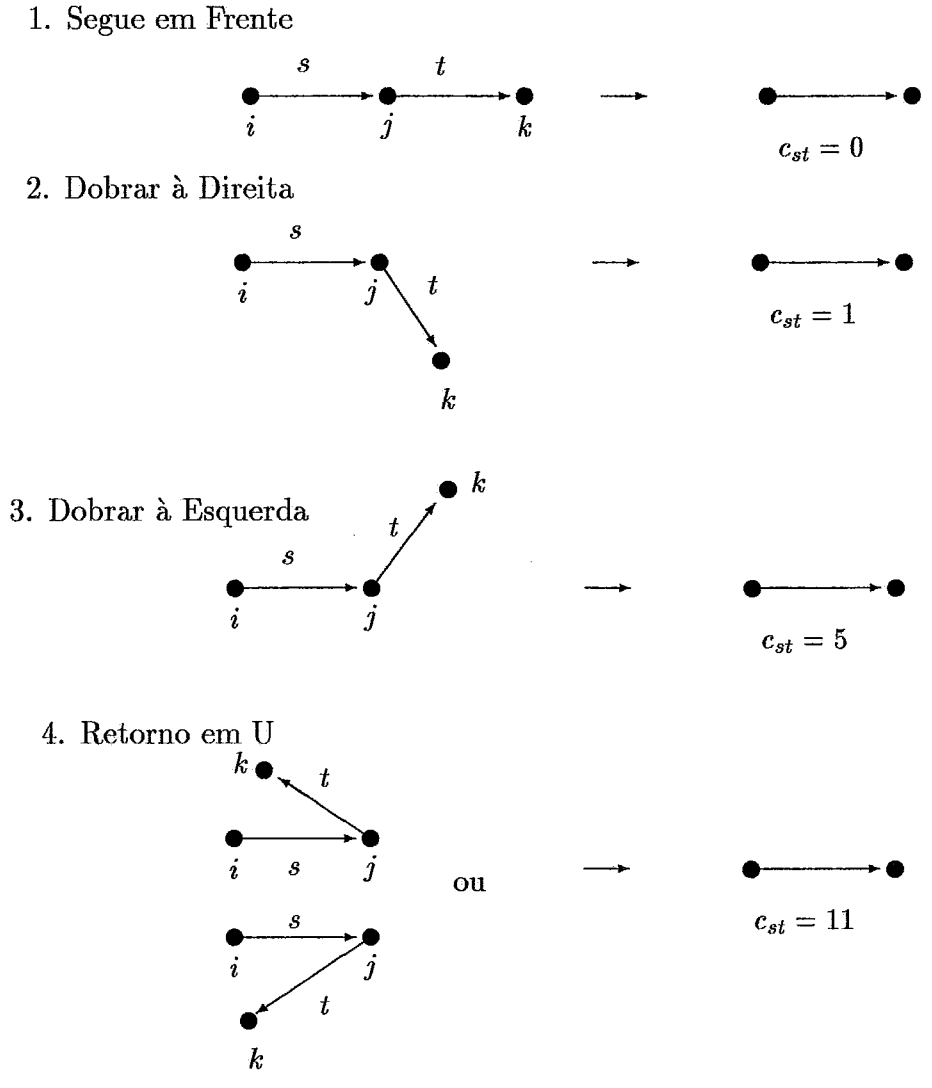


Figura VII.24: Representação de movimentação por transformação linha.

aos movimentos realizados são dados pelo esforço operacional de trafegar entre dois arcos consecutivos de G.

Na figura VII.24 tem-se representado os movimentos possíveis entre dois arcos de G, e a equivalente fomulação de transformação linha do movimento.

Os movimentos são dados de tal modo que a representação da direção de retorno e seu custo medem uma forma de condução ao longo de um percurso em um grafo euleriano. O ângulo formado entre arcos subsequentes determina para que lado foi executado o movimento, para que seja percebido corretamente o tipo do movimento realizado é preciso que a implementação tenha o cuidado de informar de algum modo a formação deste ângulo. O critério de movimentação dependerá fortemente dos critérios angulares. Na tabela VII.5 tem-se a especificação funcional

da formação do ângulo no que concerne os tipos de movimento realizados.

Tabela VII.5: Limites angulares correspondentes à movimentação de veículos entre ligações sucessivas da rota euleriana.

Movimento	Limites dos Ângulos
Seguir em Frente	$-15 \leq \theta \leq 15$
Dobrar à Direita	$-165 < \theta < -15$
Dobrar à Esquerda	$15 < \theta < 165$
Retorno em U	$165 < \theta \leq 195$

Se G é um grafo f-conexo, o percurso euleriano em G será um caminho hamiltoniano em T_G, se este é mínimo, então o esforço da rota também é mínimo.

Exemplificando, admitindo-se o grafo euleriano direcionado da figura VII.25, onde os arcos estão rotulados na ordem que aparecem na rota euleriana, a matriz de custos de T_G será dada conforme o grafo da figura VII.26, e da tabela VII.6.

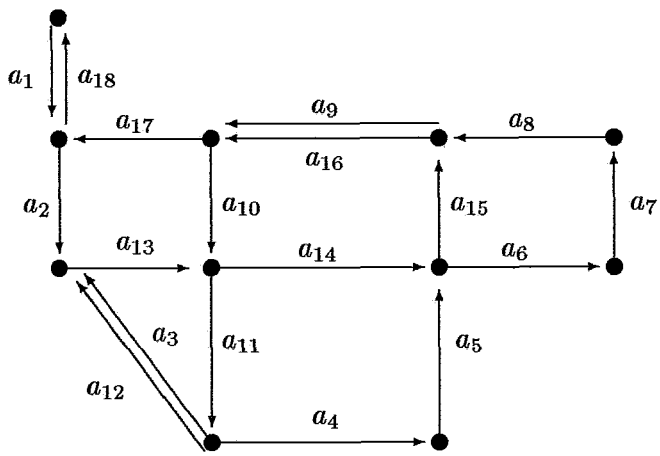


Figura VII.25: Grafo exemplo de um circuito euleriano direcionado.

Aplicando-se um algoritmo do PCV do caminho hamiltoniano entre a_1 e a_{18}, tem-se o circuito euleriano, pela definição 7.6. Cuja solução será dada por:

$$a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow a_4 \rightarrow a_5 \rightarrow a_{15} \rightarrow a_{16}$$

$$\rightarrow a_{10} \rightarrow a_{11} \rightarrow a_{12} \rightarrow a_{13} \rightarrow a_{14} \rightarrow a_6 \rightarrow a_7 \rightarrow a_8 \rightarrow a_9 \rightarrow a_{17} \rightarrow a_{18}$$

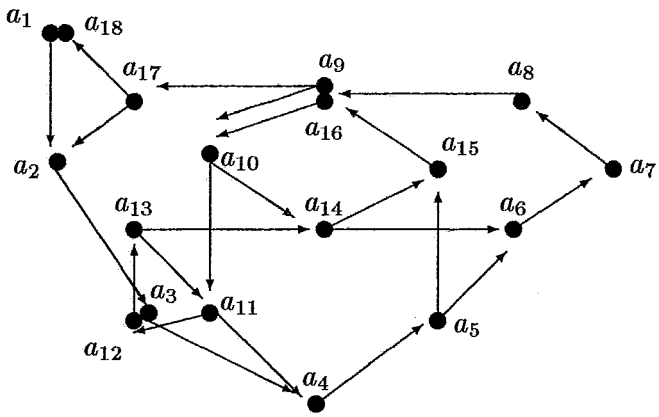


Figura VII.26: Grafo T_G referente à Transformação Linha.

Tabela VII.6: Adjacência e custos dos arcos de T_G.

(i,j)	Custo	(i,j)	Custo
1,2	0	11,4	5
1,18	11	11,12	1
2,3	5	12,3	11
2,13	5	12,13	1
3,4	5	13,11	1
3,12	11	13,14	0
4,5	5	14,6	0
5,6	1	14,15	5
5,15	0	15,9	5
6,7	5	15,16	5
7,8	5	16,10	5
8,9	0	16,17	0
8,16	0	17,2	5
9,10	5	17,18	1
9,17	0	18,1	11
10,11	0		
10,14	5		

cujo custo é de 48 pontos.

A dificuldade do procedimento está quanto à solução obtida, uma vez que ao se usar uma heurística, uma matriz de distâncias completa poderá aparecer provocando uma possível destruição do percurso. Neste caso, sugere-se o uso do algoritmo exato que possa atender a soluções de instâncias do PCV assimétrico de grande porte (> 200 vértices).

Capítulo VIII

Conclusão

Esta tese foi organizada de duas formas, a primeira buscou sempre mostrar uma parcela do estado-da-arte nos diversos assuntos abordados de modo a situar o leitor no contexto de desenvolvimento dos temas em estudo, e a segunda procurou enfatizar onde está incluida minha contribuição nas áreas de Otimização e Visualização, assim como nos Problemas de Percursos em Grafos.

A concepção do SisGRAFO foi totalmente realizada de forma independente, e a sua conceituação segue os princípios dos sistemas estado-da-arte na área. A busca, pelo usuário, de adaptação e conforto visual dos resultados, para uma melhor compreensão dos problemas relacionados, se concretizou em um sistema que vem se tornando robusto, caracterizado por ser evolutivo.

A maior dificuldade no SisGRAFO está no encaminhamento de soluções exatas para problemas semi-estruturados, porém os algoritmos heurísticos nele implementados são de expressiva qualidade e robustez.

A grande vantagem do sistema está justamente na diversidade de implementações existentes para um mesmo problema (na maioria dos problemas de rotas). Uma vez que estes são muito sensíveis a parâmetros, a mudança de estratégias no encaminhamento de soluções pode causar surpresas agradáveis ao usuário (ou modelador).

Dentro das perspectivas do sistema, mostro minha grande preocupação com o complemento de implementações de problemas importantes da literatura, com um formalismo mais adaptativo às necessidades do usuário e do projetista, e com uma fase de ambientação e geração de problemas a partir de um algoritmo geral ou um conjunto de algoritmos gerais complementares, os quais permitiriam a incorporação de novos modelos ao sistema.

No âmbito do trabalho realizado sobre os Problemas de Percursos de

Veículos, a proposta do Problema de Compras mostra-se interessante no que concerne às generalizações do problema do Caixeiro Viajante. O algoritmo apresentado, baseado em formulação dinâmica, é robusto e de boa complexidade. As aplicações deste método surgirão futuramente em problemas de manufatura flexível, uma vez que as idéias estão encaixadas a processos percorridos e manipulados por veículos automaticamente guiados (VAGs) entre células de manufatura.

Nos problemas de Percursos de Veículos em Vértices, mostrou-se uma proposta de melhoria do algoritmo de Clarke & Write, que apesar de interessante foi ofuscada pela ascenção dos métodos construtivos Metaheurísticos usando Busca TABU e *Simulated Annealing* largamente explorados nos últimos anos. Em breve, porém, estes métodos também serão superados pelos algoritmos exatos, que já estão mostrando que podem ser competitivos (em tempo de processamento) em virtude das facilidades de memória e velocidade de processamento alcançadas pelos computadores.

O Problema de Percursos de Veículos Multi-Compartimentados, foi também um dos relevantes trabalhos realizados. Apesar de já existirem métodos exatos para tratar o problema onde um compartimento só pode ser designado a um cliente (na distribuição de derivados de petróleo), a proposta de melhoria das soluções com um melhor aproveitamento da frota, usando um critério m-compartimento (mais de um cliente servido por um mesmo compartimento) supera bastante a primeira. Esta abordagem do problema é paradigmática na aplicação real uma vez que exige equipamentos extras aos veículos. Isto, porém, torna singular a abordagem dada até agora a este problema, uma vez que esta visão pode ser extendida em aplicações com múltiplos produtos e múltiplos compartimentos (*pallets*) em serviços de entrega ou coleta de artigos variados.

No Problema de Percurso em arcos, o mito do Problema do Carteiro Chinês Misto é esclarecido na maioria das considerações feitas pela literatura. O algoritmo de Sherafat é robusto, e suas idéias ajudaram diretamente na formalização da solução para o Problema do Carteiro Chinês Íngreme.

Os Problemas do Carteiro Rural e de Arcos com Demanda são na verdade os mais difíceis de serem tratados, uma vez que a construção de algoritmos 1/2-aproximativos para o PRAD (com matriz de custos que satisfaz à desigualdade triangular) é um problema $\mathcal{NP}$-Completo. Propôs-se então uma transformação polinomial do problema de arcos a um problema de nós usando transformação linha; porém, mesmo na garantia da otimalidade da solução através de um método exato, o resultado não significaria exato no retrocesso ao grafo original. A proposta de transformação linha mostrou-se robusta em diversos casos do Problema do Carteiro Rural e do PRAD, podendo também ser usada como solução para o problema de determinação da rota euleriana de esforço mínimo.

Repetiu-se aqui os resultados de Negreiros & Maculan para aplicações de transformações de arcos em nós por quebra de redes. Generalizou-se, porém, as análises realizadas com o algoritmo MARCOLIX, para os diversos casos relacionados aos problemas de percurso em arcos. O algoritmo mostrou bons resultados nas etapas de formação de soluções para o PCC e para o PRAD. Modificações simples podem ser feitas no algoritmo MARCOLIX de modo a torná-lo mais robusto, como por exemplo a troca de arcos entre rotas poderia ser uma opção. Uma outra idéia é usá-lo como gerador de população de rotas em procedimentos metaheurísticos para o PRAD.

A literatura parece não ter explorado muito as transformações de arcos em nós usando Transformações Linha no tratamento do PRAD. O uso de transformação linha é considerado aqui como um meio de conversão de um problema de percurso sobre ligações em um de percurso em vértices, utilizando procedimentos de formação de um grafo linha e aplicação de algoritmos de percurso sobre vértices. A solução obtida é então referida ao grafo original através de condições de percurso equivalente. Mostrou-se que esta técnica pode ser usada no tratamento de problemas de percurso do Carteiro Chinês e do Carteiro Rural para qualquer de suas instâncias. No PRAD também encontra-se bons resultados. Muitos trabalhos complementares podem surgir, uma vez que uma visão aproximadamente dual dos problemas de percurso do carteiro chinês e do caixeiro viajante foi estabelecida como possível.

As transformações de arcos em nós são técnicas onde alguém poderia achar que ser um retrocesso ao desenvolvimento de algoritmos para o assunto; porém, trata-se de fato de uma outra visão (um paradigma) do mesmo problema. É relevante abordar problemas de formas diferentes à grande tendência da literatura, isto faz com que se abra mais as idéias sobre os encaminhamentos das soluções e favorece a uma melhor estruturação de consciência própria e original de novas estratégias a seguir em problemas semi-estruturados.

No que se refere a SADs para Otimização em Grafos, o SisGRAFO está de fato ainda distante de ser um SAD completo. Muitos trabalhos interessantes podem surgir das idéias aqui colocadas para o SisGRAFO, dentre elas destaco: O formulador geral de problemas e o gerador de produções.

Este trabalho foi feito com uma dose excessiva de paixão sobre tudo aquilo que faço por prazer e obrigação. O meu maior regozijo está em que alguns dos resultados atingidos com o uso do SisGRAFO estão aplicados em algumas situações de planejamento reais, reduzindo drasticamente custos, melhorando o atendimento prestado, e aliviando o sofrível esforço de trabalhadores no seu dia-a-dia. Para mim esta foi, é e será a maior contribuição do SisGRAFO.

Referências Bibliográficas

[Agarwal et al 89] Y. Agarwal, K. Mathur & H.M. Salkin (1989) - "A Set-Partioning Based Exact Algorithm for the Vehicle Routing Problem" - *Networks*, vol. 19 pgs. 731-749.

[Alprin 75] B.S. Alprin (1975) - "A Simulation Approach to Solving the Snow and Ice Removal Problem in Urban Areas" - MSc Dissertation, University of Tulsa, Oklahoma.

[Altinkemer & Gavish 91] K. Altinkemer & B. Gavish (1991) - "Parallel Savings Based Heuristic for the Delivery Problem" - *Operations Research*, vol. 39 pgs. 456-469.

[Antony 65] R.N. Antony (1965) - *Planning and Control Systems: A Framework for Analysis*, Cambridge, MA: Harvard University Graduate School of Business. Prentice Hall.

[Assad & Golden 88] A. Assad & B. Golden (1988) - *Modeling and Implementation Issues in Vehicle Routing* - Vehicle Routing Methods and Studies, North Holland.

[Assad et al 1987] A.A Assad, W.L. Pearn & B. L. Golden (1987) - "The Capacitated Chinese Postman Problem: Lower Bounds and Solvable Cases" - *American Journal of Mathematical and Management Sciences*, vol 7, pgs. 63-88

[Baker & Schaffer 86] E. Baker & J. Schaffer (1986) - "Solution Improvement Heuristics for the Vehicle Routing and Scheduling Problem with Time Window" - *American J. Math. Management Science*, vol. 6 pgs. 261-300.

[Balakrisnan 93] N. Balakrisnan (1993) - "Simple Heuristics for the Vehicle Routeing Problem with Soft Time Window" - *Journal of the Operations Research Society*, vol. 44(3) pgs. 279-287.

[Balas 89] E. Balas (1989) - "The Prize-Collecting Traveling Salesman Problem" - *Networks*, vol. 19, pgs. 621-636.

[Balas & Martin 85] E. Balas & G. Martin (1985) - "ROLL-A-ROUND: Software Package for Scheduling the Rounds of a Rolling Mill" - Copyright Balas and Martin Associates, 104 Maple Heights Road, Pittsburgh.

[Balinski & Quandt 64] M.L. Balinski & R.E. Quandt (1964) - "On an Integer Program for a Delivery Problem" - *Operations Research*, vol. 12(2) pgs. 300-304.

[Ball et al 83] M. Ball, B. Golden, A. Assad & L. Bodin (1983) - "Planning for Truck Fleet Size in the Presence of Common-Carrier Option" - *Decision Science*, vol. 14 pgs. 103-120.

[Backer & Schaffer 86] E. Backer & J. Schaffer (1986) - "Computacional Experience with Branch Exchange Heuristics for Vehicle Routing Problems with Time Window Constraints" - *Am. Journal of Math. and Manag. Sciences*, vol.6(3/4) pgs. 261-300.

[Barahona 90] F. Barahona (1990) - "On Some Applications of the Chinese Postman Problem" - *Algoritms and Combinatorics*, vol. 9, *Paths, Flows and VLSI-Layout*, B. Korte, L. Lovasz, H. J. Promel, e A. Schriver (eds.) Spring-Verlag, Berlin.

[Bausch et al 91] D.O. Bausch, G.G. Brown & D. Ronen (1991) - "Elastic Set Partitioning: A Powerful Tool for Scheduling Transportation of Oil and Gas" - *Advances in Operations Research in the Oil and Gas Industry*, M. Breton, G. Zaccour (Ed.) and Technip, Paris.

[Bazaraa et al 90] M.S. Bazaraa, J.J. Jarvis & H.D. Sherali (1990) - *Linear Programming and Network Flow* - New York, John Wiley & Sons, Singapore, 2nd. Ed.

[Belardo et al 82] S. Belardo, P. Duchessi & J. Seagle (1982) - "Microcomputer Graphics in Support of Vehicle Fleet Routing" - *Interfaces*, vol. 15 pgs. 84-92.

[Belenguer & Benavent 91] J.M. Belenguer & E. Benavent (1991) - "Polyhedral Results on the Capacitated Arc Routing Problem" - Working Paper, Departamento de Estadística e Investigación Operativa, Universidad de Valencia.

[Bell 85a] P. Bell (1985) - "Visual Interactive Modeling as an OR Technique", *Interfaces*, Vol. 15(4), pgs. 26-33.

[Bell 85b] P. Bell (1985) - "Visual Interactive Modeling in Operations Research: Success and Opportunities", *J. Op. Res. Society*, Vol. 36, pgs. 975-982.

[Bell 86] P. Bell (1985) - "Visual Interactive Modeling in 1986", pgs. 1-12, *Recent Developments in Operations Research*, V. Belton & O'Keefe (eds.), Paragon Press, Oxford, UK.

[Bell 91] P. Bell (1991) - "Visual Interactive Modelling: The Past, the Present, and the Prospects", *EJOR*, Vol. 54, pgs. 274-286.

[Bell et al 83] W. Bell, L. Dalberto, M. Fisher, A. Greenfield, R. Jaikumar, R. Mack & P. Prutzman (1983) - "Improving the Distribution of Industrial Gases with an On-Line Computerized Routing and Scheduling System" - *Interfaces*, vol. 13 pgs. 4-23.

[Bell & O'Keefe 87] P. Bell & R.M. O'Kneefe (1987) - "Visual Interactive Simulation: History, Recent Developments, and Major Issues", *Simulation*, Vol. 49(3), pgs. 109-116.

[Belton & Elder 91] V. Belton & M. Elder (1991) - "Editorial - Visual Interactive Modelling", *EJOR*, Vol. 54 pg. 273.

[Benders 62] J.F. Benders (1962) - "Partitioning Procedures for Solving Mixed-Variables Programming Problems" - *Numerische Mathematik*, vol. 4 pgs. 238-252.

[Bodin et al 83] L. Bodin, B. Golden, A. Assad & M. Ball (1983) - "Routing and Scheduling of Vehicle and Crews - The state of art." - *Comp. & Op. Res.* vol. 10(2) pgs. 63-211.

[Bodin & Beltrami 74] L.D. Bodin & E.J. Beltrami (1974) - "Networks and Vehicle Routing for Municipal Waste Collection" - *Networks*, Vol. 4 pgs. 65-94

[Bodin & Krush 78] L.D. Bodin & S.J. Kursh (1978) - "A Computer-Assisted System for the Routing and Scheduling of Street Sweepers" - *Operations Research* vol 26, pgs. 525-537

[Bodin & Krush 79] L.D. Bodin & S.J. Kursh (1979) - "A Detailed Description of a Computer System for the Routing and Scheduling of Street Sweepers" - *Comp. & Op. Research* vol 6, pgs. 181-198

[Bodin et al 89] L.B. Bodin, G. Fagin, R. Welebny, & J. Greenberg (1989) - "The Design of a Computerized Sanitation Vehicle Routing and Scheduling System for the Town of Oyster Bay, New York", *Computers & Op. Res.*, Vol. 16(1) pgs. 45-54

[Bonavent et al 90] E. Bonavent, V. Campos, A. Corberán e E. Mota (1990) - "The Capacitated Arc Routing Problem. A Heuristic Algorithm", *Quaderns d' Estadística, Sistemes, Informática i Investigacion Operativa (QÜESTIIÓ)*, Vol. 14, pgs. 107-122.

[Bonavent et al 92] E. Bonavent, V. Campos, A. Corberán e E. Mota (1992) - "The Capacitated Arc Routing Problem" - *Networks*, vol 22, pgs. 669-690.

[Brown & Graves 81] G. Brown & G. Graves (1981) - "Real-Time Dispatch of Petroleum Tank Trucks" - *Management Science*, vol. 27 pgs. 19-31.

[Brown et al 87] G. Brown, C. Ellis, G. Graves & D. Ronen (1987) - "Real Time Wide Area Dispatch of Mobil Tank Trucks" - *Management Science*, vol. 27 pgs. 19-32.

[Burkard & Derigs 80] R. E. Burkard & U. DERIGS (1980) - *Assignment and Matching Problems: Solution Methods with FORTRAN programs* - Lecture Notes in Economics and Mathematical Systems, No. 184, Spring Verlag, Berlin.

[Burrows 88] W. Burrows - "The Vehicle Routing Problem with LoadSplitting" - *24th Annual Conference of the Operational Research Society of New Zealand*, pgs. 33-38.

[Busch 91] I.K. Busch (1991) - *Vehicle Routing on Acyclic Networks*, PhD Dissertation, The Johns Hopkins University.

[Campello & Maculan 94] Campello, R. & Maculan, N. (1991) - *Algoritmos e Heurísticas Desenvolvimento e Avaliação de Performace*, Ed. da Universidade Federal Fluminense (EDUFF/FURNAS) - Rio de Janeiro.

[Carpaneto et al 89] G. Carpaneto, M. Fischetti & P. Toth (1989) - "New Lower Bounds for the Symmetric Travelling Salesman Problem" - *Mathematical Programming*, vol. 45 pgs. 233-254.

[Carvalho & Ochi 92] J.L. Carvalho & L.S. Ochi (1992) - "Método de Decomposição em Pétalas Associado a um Algoritmo Heurístico Construtivo e de Busca Local para um Problema de Percursos com Janelas de Tempo", *Relatório Técnico - GMC/UFF: 14*, Universidade Federal Fluminense/RJ.

[Carpaneto et al 89] G. Carpaneto, M. Dell' Amico, M. Fischetti & P. Toth (1998) - "A Branch and Bound Algorithm for the Multiple Depot Vehicle Scheduling Problem" - *Networks*, vol. 19 pgs. 531-548.

[Casco et al 88] D. Casco, B. Golden & E. Wasil (1988) - "Vehicle Routing with Backhauls: Models, Algorithms, and Case Studies" - *Vehicle Routing: Methods and Studies*, B. Golden and A. Assad (Ed.) North-Holland, pgs. 127-147.

[Chao et al 93] I-M. Chao, B.L. Golden & E. Wasil (1993) - "A New Heuristic for the Multi-Depot Vehicle Routing Problem that Improves upon Best-Known Solutions", *American Journal of Mathematical and Management Sciences*, Vol. 13(3), pgs. 371-406.

[Chapleau et al 84] L. Chapleau, J.A. Ferland, G. Lapalme & J.M. Rousseau - "A Parallel Insert Method for The Capacitated Arc Routing Problem" - *Operations Research Lettres*, Vol. 3, pgs. 95-99.

[Chinnek & Dravnieks 89] J.W. Chinneck & E.W. Dravnieks (1989) - "Locating Minimal Infeasible Constraint Sets in Linear Programming", *ORSA Journal on Computing*, Vol. 3(2), pgs. 157-168.

[Christofides 73] N. Christofides (1973) - "The Optimal Traversal of a Graph" - Omega, Vol. 1 pgs. 719-732

[Christofides 76] N. Christofides (1976) - "Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem." - Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA.

[Christofides 78] N. Christofides (1978) - *Graph Theory, An Algorithm Approach*, Academic Press, London

[Christofides 85] N. Christofides (1985) - "Vehicle Scheduling and Routing", *12th International Symposium on Mathematical Programming*, Cambridge, MA.

[Christofides & Eilon 69] N. Christofides & S. Eilon (1969) - "An Algorithm for the Vehicle Dispatching Problem", *Operational Research Quaterly*, Vol. 20, pgs. 309-318.

[Christofides et al 81] N. Christofides, V. Campos, A. Coberán & E. Mota (1981) - "An Algorithm for the Rural Postman Problem" - Imperial College Report, IC.O.R.81.5.

[Christofides et al 86] N. Christofides, V. Campos, A. Coberán & E. Mota (1986) - "An Algorithm for the Rural Postman Problem on a Directed Graph" - *Mathematical Programming Study*, vol 26 pgs. 155-166.

[Christofides & Beasley 84] N. Christofides & J. Beasley (1984) - "The Period Routing Problem", *Networks*, Vol. 14, pgs. 237-256.

[Christofides, Mingozzi & Toth 79] N. Christofides. A. Mingozzi & P. Toth (1979) - "The Vehicle Routing Problem", *Combinatorial Optimization*, Wiley, Chichester, pgs. 315-338.

[Christofides et al 81a] N. Christofides. A. Mingozzi & P. Toth (1981) - "Exact Algorithms for the Vehicle Routing Problem Based on Spanning Tree and Shortest Path Relaxations", *Mathematical Programming*, Vol. 20, pgs. 255-282.

[Christofides et al 81b] N. Christofides. A. Mingozzi & P. Toth (1981) - "State Space Relaxation Procedures for the Computation of Bounds to Routing Problems", *Networks*, Vol. 11, pgs. 145-164.

[Clarke & Write 64] G. Clarke & J.W. Write (1964) - "Scheduling of Vehicles From a Central Depot to a Number of Delivery Points", *Op. Research*, Vol. 12, pgs. 568-581.

[Cornuéjols & Harche 93] G. Cornuéjols & F. Harche (1993) - "Polyhedral Study of the Capacitated Vehicle Routing Problem", *Mathematical Programming*, Vol. 60, pgs. 21-52.

[Cullen et al 81] F. Cullen, J. Jarves & Ratliff (1981) - "Set Partioning Based Heuristics for Interactive Routing" - *Networks*, vol. 11 pgs. 125-144.

[Cook & Alprin 76] T.M. Cook & B.S. Alprin (1976) - "Snow and Ice Removal in an Urban Enviroment" - *Management Science* vol 23, pgs. 227-234

[Corberán & Sanchis 91] A. Corberán & J.M. Sanchis (1976) - "A Poliedral Approach to the Rural Postman Problem" - *Working Paper*, Departamento de Estadística i Investigación Operativa, Universidade de Valencia, Spain

[Croes 54] A. Croes (1954) - "A Method for Solving Traveling Salesman Problems" - *Operations Research* vol 2, pgs 393-410

[Cross 36] H. Cross (1936) - "Analysis of Flow in Networks of Conduits of Conductors" - *Bull* vol 286, University of Illinois Engeneering Experimental Station, Urbana, Illinois.

[Dantzig et al 54] G.B. Dantzig, D.R. Fulkerson & S.M. Johnson (1954) - "Solution of a large-scale Traveling Salesman Problem" - *Operations Research*, vol. 2 pgs. 393-410.

[Dantzig & Ramser 59] G.B. Dantzig & J.H. Ramser (1959) - "The Trucking Dispatching Problem", *Management Science*, Vol. 6(1), pgs. 80-91.

[Dao et al 86] M. Dao, M. Habib, J.P. Richard & D. Tallot (1986) - "CABRI: An Interactive System for Graph Manipulation", Tinhofer & Schmidth (Eds.) *Graph-Theoretic Concepts in Computer Science*, Springer-Verlag, Berlin.

[Dean et al 92] N. Dean, M. Mevenkamp & C.L. Monma (1980) - "NETPAD, An Interactive Graphics System for Network Modeling and Optimization", pgs. 231-243, *Computer Science and Operations Research: New Developments in Their Interfaces*, Balci, Sharda & Zenios (Ed.), Paragon Press, Oxford,UK.

[Deif & Bodin 84] I. Deif & L. Bodin (1984) - "Extension of the Clarke and Write Algorithm for Solving the Vehicle Routing Problem with Backhauls", *Proceedings of Babson Conference on Software Uses in Transportation and Logistics Management*, pgs. 75-96, A.E. Kidder (Ed.) Babson Park, Massachussets.

[Delgrande 80] J.P. Delgrande (1980) - "A Graph-Theoretic Language Extension for an Interactive Computer Graphics Environment", *Computer and Graphics*, Vol. 8, pgs. 13-22.

[Desrosiers et al 86] J. Desrosiers, J.A. Ferland, J-M. Rousseau, G. Lapalme & L. Chapleau (1986) - "TRANSCOL: A Multi-Period School Bus Routing and Scheduling System" - *Studies Management Science*, vol. 22 pgs. 47-71

[Desrosiers et al 84] J.Desrosiers, M. Desrochers & F. Soumis (1984) - "Routing with Time Windows by Column Generation", *Networks*, Vol. 14, pgs. 545-565.

[Desrosiers et al 88] J.Desrosiers, M. Sauvé & F. Soumis (1988) - "Lagrangean Relaxation Methods for Solving the Minimum Fleet Size Multiple Travelling Salesman Problem with Time Windows", *Management Science*, Vol. 34(8), pgs. 1005-1022.

[Desrochers et al 92] M. Desrochers, J. Desrosiers & M. Solomon (1992) - "A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows", *Operations Research*, Vol. 40(2), pgs. 342-354.

[Dijkstra 59] Dijkstra, E. (1959) - "A Note on Two Problems in Connection with Graphs" - *Numer. Math.*, Vol. 1 pgs. 269-271.

[Dinic 70] E.A. Dinic (1970) – "Algorithm for Solution of a Problem of Maximal Flow in a Network with Power Estimation", *Soviet Math. Dokl.* vol. 11, pgs 1277–1280.

[Donohue 95] J. Donohue (1995) - "NETWK, Problem-solving tool for network and ehanced network models", *OR/MS Today*, december, pgs. 46-49.

[Dror 83] M. Dror (1983) - "The Inventory Routing Problem", - *Ph.D Thesis*, College of Business and Management, University of Maryland.

[Dror 93] M. Dror (1993) - "Modeling Vehicle Routing with Uncertain Demands as a Stochastic Program: Propeties of the Corresponding Solution", - *EJOR*, vol. 64, pgs. 432-441.

[Dror & Levi 86] M. Dror & L. Levy (1986) - "A Vehicle Routing Improvement Algorithm Comparison of a Greedy and a Matching Approach for Inventory Routing", *Computers & Op. Res.*, Vol. 13, pgs. 33-45.

[Dror et al 94] M. Dror, G. Laporte & P. Trudeau (1994) - "Vehicle Routing with Split Deliveries", *Discrete Applied Mathematics*, Vol. 50, pgs. 239-254.

[Dror et al 87] M. Dror, H. Stern & P. Trudeau (1987) - "Postman Tour on a Graph with Precedence Relation on Arcs" - *Networks*, vol. 17 pgs. 283-294

[Dror & Trudeau 90] M. Dror & P. Trudeau (1990) - "Split Delivery Routing", *Naval Research Logistics*, Vol. 37, pgs. 383-402.

[Easa 85] S.M. Easa (1985) - "Shortest Route Algorithm with Movement Prohibitions", *Transportation Research*, Vol. 19B, pgs. 197-208.

[Edmonds & Johnson 73] J. Edmonds & E. Johnson (1973) - "Matching, Euler tours and the Chinese Postman Problem" - *Mathematical Programming*, vol. 5 pgs. 88-124

[Eglese & Li 92] R.W. Eglese & L.Y.O Li (1992) - "Efficient Routing for Winter Gritting" - *Journal of ORSA* 43, pgs. 1031-1034

[Eglese & Murdock 91] R.W. Eglese & H. Murdock (1991) - "Routing Road Sweepers in a Rural Area" - *Journal of ORSA* 42, pgs. 281-288

[Eilon et al 71] S. Eilon, C.D.T Watson-Gandy & N. Christofides (1971) - *Distribution Management: Mathematical Modelling and Practical Analysis*, Griffin, London.

[Eiselt et al 95i] H.A. Eiselt, M. Gendreau & G. Laporte (1995) - "Arc Routing Problems, Part I: The Chinese Postman Problem" - *Operations Research*, vol. 43(2) pgs. 231-242

[Eiselt et al 95ii] H.A. Eiselt, M. Gendreau & G. Laporte (1995) - "Arc Routing Problems, Part II: The Rural Postman Problem" - *Operations Research*, vol. 43(3) pgs. 399-414

[Etezadi & Beasley 83] T. Etezadi & J. Beasley (1983) - "Vehicle Fleet Composition", *J. Op. Res. Society*, Vol. 34, pgs. 87-91.

[Euler 1736] L. Euler (1736) - "Solutio Problematics ad Geometrian Situs Pertinentis" - Commentarii Academae Scientarum, Petropolitanae 8, pgs. 128-140

[Federgruen & Zipkin 84] A. Federgruen & P. Zipkin (1984) - "A Combined Vehicle Routing and Inventory Allocation Problem", *Op. Research*, Vol. 32, pgs. 1019-1037.

[Fiacco 90] A.V. Fiacco (ed.) (1990) - *Annals of Operational Research*, Vol. 27, pgs. 1-4, Special Issue on Optimization with Data Pertubations.

[Fichetti & Toth 88] M. Fichetti & P. Toth (1988) - "An Additive Approach for the Optimal Solution of the Prize Collecting TSP", em Bruce Golden and A. Assad (ed.), *Vehicle Routing: Methods and Studies*, North-Holland, Amsterdam, pgs. 319-343.

[Finke et al 84] G. Finke, A. Claus & E. Gunn (1984) - "A Two-Commodity Network Flow Approach to the Traveling Salesman Problem", *Congressus Numerantium*, vol 41 pgs. 167-178.

[Fisher 86] M.L. Fisher (1986) - "Interactive Optimization", *Annals of Operational Research*, Vol. 5, pgs. 541-556.

[Fisher 94] M.L. Fisher (1994) - "Optimal Solution of Vehicle Routing Problems Using Minimum K-Trees", *Op. Research*, Vol. 42, pgs. 626-642.

[Fisher & Jaikumar 81] M.L. Fisher & R. Jaikumar (1981) - "A Generalized Assinment Heuristic for Vehicle Routing", *Networks*, Vol. 11, pgs. 109-124.

[Fisher et al 91] M.L. Fisher, K. Jörnsten, & O. Madsen (1991) - "Vehicle Routing with Time Windows", *Preliminary Results*, Technical Report 4/1991, IMSOR, The Technical University of Denmark.

[Ford & Fulkerson 62] L.R. Ford & D.R. Fulkerson (1962) - *Flows in Networks.* Princeton University Press, Princeton NJ (USA).

[Foster & Ryan 76] B.A. Foster & D.M. Rayan (1976) - "An Integer Programming Approach to the Vehicle Scheduling Problem", *Op. Res. Quarterly*, Vol. 27, pgs. 367-384.

[Franz & Woodmansee 90] L. Franz & J. Woodmansee (1990) - "Computer-Aided Truck Dispatching Under Conditions of Product Price Variance with Limited Supply" - *J. Business Logistics*, vol. 11 pgs. 127-139.

[Frederickson 79] G. Frederickson - "Approximation Algorithms for Some Postman Problems" - *Journal of ACM*, Vol. 26, pgs. 538-554

[Frederickson et al 78] G. Frederickson, M. Hecht & C. KIM (1978) - "Approximation Algorithms for some Routing Problems" - *SIAM Journal of Computers*, Vol. 7(2) pgs. 178-193

[Garey & Johnson 79] M.R Garay & D.S. Johnson (1979) - *Computers and Intractability: A guide to the theory of NP-Completeness* - Ed. Freeman, San Francisco.

[Garfinkel 70] R. Garfinkel (1970) - "An Improved Algorithm for the Bottleneck Assignment Problem" - *Working Paper, no. 7003*, College of Business and Administration, University of Rochester, New York.

[Galdioso & Palleta 92] M. Galdioso & G. Paletta (1992) - "A Heuristic for the Periodic Vehicle Routing Problem" - *Transportation Science*, Vol. 26(2) pgs. 86-92.

[Gavish & Graves 78] B. Gavish & S. Graves (1978) - *The Traveling Salesman Problem and Related Problems* - Working Paper, Graduate School of Management, University of Rochester.

[Gavish & Srikanth 86] B. Gavish & K.N. Srikanth (1986) - "An Optimum Solution Method for Large-Scale Multiple Traveling Salesman Problems" - *Operations Research* vol. 34, pgs. 698-717.

[Gaskell 67] T.J. Gaskell (1967) - "Bases for Vehicle Fleet Scheduling" - *Op. Res. Quarterly*, Vol. 18 pgs. 281-295

[Gelinas 92] É. Gelinas (1992) - "Le Probléme du Postier Chinois avec Contraints Générales de Préséance" - MSc.A. Dissertation, École Polytechnique de Montréal, Canada.

[Gendreau et al 90] M. Gendreau, G. Laporte & Y. Zhao (1990) - "The Windy Postman Problem on General Graphs", *CRT*, report no. 698, Montreal, Canada.

[Gendreau et al 90] M. Gendreau, A. Hertz & G. Laporte (1992) - "New Insertion and Post-Optimization Procedures for the Traveling Salesman Problem." - *Operations Research* vol. 40(6), pgs. 1086-1094.

[Gendreau et al 94] M. Gendreau, A. Hertz & G. Laporte (1994) - "A Tabu Search Heuristic for the Vehicle Routing Problem" - *Management Science*, Vol. 40(10) pgs. 1272-1290.

[Geoffrion 87] A.M. Geoffrion (1987) - "An Introduction to Structured Modelling", *Management Science*, Vol. 33(5), pgs. 547-588.

[Geoffrion 89] A.M. Geoffrion (1989) - "The Formal Aspects for Structured Modeling", *Op. Research*, Vol. 37(1), pgs. 30-51.

[Gheysens et al 86] F. Gheysens, B.L. Golden & A. Assad (1986) - "A Comparison of Thechnics for Solving the Fleet Size and Mix Vehicle Routing", *OR Spektrum*, Vol. 6, pgs. 207-216.

[Gillett & Miller 74] B.E Gillett & L.R. Miller (1974) - "A Heuristic Algorithm for the Vehicle Dispatch Problem" - *Operations Research*, Vol. 22 pgs. 340-349.

[Gillet & Johnson 76] B. Gillet & J. Johnson (1976) - "Multi-Terminal Vehicle-Dispatch Algorithm" - *OMEGA*, Vol. 4(6) pgs. 711-718.

[Glover 89] F. Glover (1989) - "Tabu Search - Part I" - *ORSA Journal of Computing* vol. 1, pgs. 190-209.

[Glover 90a] F. Glover (1990) - "Tabu Search - Part II" - *ORSA Journal of Computing* vol. 2, pgs. 4-32.

[Glover 90b] F. Glover (1990) - "Tabu Search: A Tutorial" - *Interfaces*, Vol. 20 pgs. 340-349.

[Glover et al 77] F. Glover, D. Klingman & N. Phillips (1977) - "NETFORM Concept: A More Effective Model Form and Solution Procedure for Large Scale Nonlinear Problems", *National Technical Information Service*, US Department of Commerce, Springfield, VA.

[Goetschslckx & Jacobs-Blecha 89] M. Goetschslckx & C. Jacobs-Blecha (1989) - "The Vehicle Routing Problem with Backhauls" - *EJOR*, Vol. 42 pgs. 39-51.

[Golden & Assad 86] B.L. Golden & A. Assad (1986) - "Perspectives on Vehicle Routing: Exciting New Developments" - *Operations Research*, Vol. 34(5) pgs. 803-810.

[Golden & Assad 88] B.L. Golden & A. Assad (1986) - *Vehicle Routing: Methods and Studies* - Studies in Management Science and Systems, Vol. 16, Ed. Golden & Assad, North-Holland, Amsterdam.

[Golden et al 77] B.L. Golden, T.L. Magnanti & H.Q. Nguyen (1977) - "Implementing Vehicle Routing Algorithms" - *Networks*, Vol. 7 pgs. 113-148.

[Golden et al 84a] B. Golden, A. Assad. & R. Dahl (1984) - "Analysis of Large Scale Vehicle Routing with Inventory Component" - *Large Scale Systems*, Vol. 7 pgs. 181-190.

[Golden et al 84b] B. Golden, A. Assad, L. Levy & F. Gheysens (1984) - "The Fleet Size and Mix Vehicle Routing Problem" - *Computers & Op. Res.*, Vol. 11(1) pgs. 49-66

[Golden et al 86] B. Golden, L. Bodin & T. Goodwin (1986) - "Microcomputer-Based Vehicle Routing and Scheduling Software" - *Computers & Op. Res.*, Vol. 13 pgs. 277-285.

[Golden et al 81] B. Golden, L. Levy & R. Dahl (1981) – "Two Generalizations of the Travelling Salesman Problem", *Omega*, Vol. 19(4)

[Golden et al 87] B. Golden, L. Levy & R. Vohra (1987) - "The Orientering Problem" - *Naval Research Logistics*, Vol. 34 pgs. 307-318.

[Golden et al 88] B. Golden, Q. Wang. & L. Liu (1988) - "A Multifaced Heuristic for the Orienteering Problem" - *Naval Research Logistics*, Vol. 35 pgs. 359-366.

[Golden et al 83] B. Golden, J.S. DeArmon & E.K. Baker (1983) - "Computational Experiments with Algorithms for a Class of Routing Problems" - *Computers & Op. Res*, Vol. 10, pgs. 47-49

[Golden & Stewart 85] B. Golden & W.R. Stewart (1985) - "Empirical Analysis of heuristics" - Lawler, Lenstra, Rinnooy Kan, Shmoys (Eds) - *The Traveling Salesman Problem. A Guided tour of Combinatorial Optimization*, Wiley, Chichester pgs. 207-249.

[Golden & Wasil 87] B. Golden & E. Wasil (1987) - "Computerized Vehicle Routing in the Soft Drink Industry" - *Op. Research*, Vol. 35(1) pgs. 6-17.

[Golden & Wong 81] B. Golden & R. Wong (1981) - "Capacitated Arc Routing Problems" - *Networks*, Vol. 11 pgs. 305-315

[Golden & Wong 92] B.L. Golden & R.T. Wong (1992) - "Vehicle Routing by Land, Sea and Air" - *Interfaces*, Vol. 22(3) pgs. 1-4.

[Grant & Weiner 86] J.W. Grant & Weiner (1986) - "Factors to Consider in Choosing a Graphically Animated Simulation System", *Industrial Engeneering*, Vol. 18.

[Götler 87] H. Götler (1987) - "Graph-Grammars and Diagram Editing", Ehrig, Nagl, Rozenberg & Rosenfeld (Eds) *Graph Grammars and Their Application to Computer Science*, Springer-Verlag, Berlin, pgs. 130-142.

[Götler 89] H. Götler (1989) - "Graph-Grammars A New Paradigm for Implementing Visual Languages", Deshowitz (Ed) *Rewriting Techniques and Applications to Computer Science*, (Lecture Notes in Computer Sciencem 355) Springer-Verlag, Berlin.

[Grötschel & Padberg 79a] M. Grötschel & M.W. Padberg (1979) - "On the Symmetric Traveling Salesman Problem I." - *Mathemmatical Programming* vol. 16 pgs. 265-280.

[Grötschel & Padberg 79b] M. Grötschel & M.W. Padberg (1979) - "On the Symmetric Traveling Salesman Problem II - Lifting theorems and facets." *Mathemmatical Programming* vol. 16 pgs. 281-302.

[Grötschel & Holland 91] M. Grötschel & O. Holland (1991) - "Solution of Large Scale Symmetric Traveling Salesman Problem." *Mathemmatical Programming* vol. 51 pgs. 141-202.

[Grötschel & Win 88] M. Grötschel & Z. Win (1988) - "On the Windy Postman Polyhedron", *SDF*, report no. 75, Augsburg, Germany.

[Grötschel & Win 92] M. Grötschel & Z. Win (1992) - "A Cutting Plane Algorithm for the Windy Postman Problem", *Mathematical Programming*, vol 55, pgs 339-358.

[Guedes 90] L. F. Guedes (1990) - "Teoria Algébrica dos Grafos"- *2a. Semana de Matemática, Estatística e Computação e 1o. Encontro Regional de Matemática Aplicada e Computacional*, (notas de minicurso), Universidade Federal Fluminense.

[Haslam & Write 91] E. Haslam & J.R. Write (1991) - "Application of Routing Technologies to Rural Snow and Ice Control", *Transportation Research Record* no. 1304, pgs. 202-211.

[Heierholzer 1873] C. Heierholzer (1873) - "Uber die Moglichkeit, einen Linienzung ohne Wiederholing und ohne Unterbrechung zu umfahren" - Mathematische Annalen IV, pgs. 30-32.

[Held & Karp 70] M. Held & R.M. Karp (1970) - "The Traveling Salesman Problem and Minimum Spanning Trees." *Operations Research* vol. 18 pgs. 1138-1162.

[Held & Karp 71] M. Held & R.M. Karp (1971) - "The Traveling Salesman Problem and Minimum Spanning Trees - Part II." *Mathematical Programming* vol. 1 pgs. 6-25.

[Hjorring 95] C.A. Hjorring (1995) - *The Vehicle Routing Problem and Local Search Metaheuristics* - Doctoral Dissertation, University of Auckland.

[Hiquebran et al 94] D. Hiquebran, A. Alfa, J. Shapiro, D. Gittoes (1994) - "A Revised Simulated Annealing and Cluster-Frist Route Second Algorithm Applied to the Vehicle Routing Problem" - *Engineering Optimization*, Vol. 22 pgs. 77-107.

[Hurrion 86] R.D. Hurrion (1986) - "Visual Interactive Modeling", *EJOR*, Vol. 23, pgs. 281-287.

[Hurrion & Secker 78] R.D. Hurrion & R.J.R. Secker (1996) - "Visual Interactive Simulation: An Aid to Decision Making", *Omega*, Vol. 6, pgs. 419-426.

[Itai & Rodeh 78] A. Itai & M. Rodeh (1978) - "Covering a Graph by Circuits" - *Lecture Notes in Computer Science*, vol. 62 pgs. 289-299.

[Itai et al 81] A. Itai, J. Lipton, C.H. Papadimitriou & M. Rodeh (1981) -"Covering a Graph by Simple Circuits" - *SIAM Journal on Computing*, vol. 10 pgs. 746-750.

[Jansen & Barnes 90] P.A. JANSEN & J.W. BARNES (1990) *Network Flow Programming*, John Wiley & Sons, New York

[Jaw et al 86] J. Jaw, A. Odoni, H. Psarafis & N. Wilson (1986) - "A Heuristic Algorithm for the Multi-Vehicle Advance-Request Dial-a-Ride Problem with Time Windows" - *Transp. Research*, Vol. 20B pgs. 243-257.

[Johnson 90] D.S. Johnson (1990) - "Local Optimization and the Traveling Salesman Problem", M.S. Petersen Ed., *Preceedings of the 17th International Colloquium on Automata, Languages and Programming* Lecture Notes in Computer Science, Springer-Verlag, Berlin, pgs. 446-461.

[Jones 90] C.V. Jones (1990) - "An Introduction to Graph-Based Modeling Systems: Part I Overview", *ORSA Journal on Computing*, Vol. 2(2), pgs. 136-151.

[Jones 91] C.V. Jones (1991) - "An Introduction to Graph-Based Modeling Systems: Part II Graph-Grammars and the Implementation", *ORSA Journal on Computing*, Vol. 3(3), pgs. 180-206.

[Jones 92] C.V. Jones (1992) - "Attributed Graphs, Graph-Grammars, and Structured Modeling", *ORSA Journal on Computing*, Vol. 38, pgs. 281-324.

[Jones 93] C.V. Jones (1993) - "An Integrated Modeling Environment Based on Attributed Graphs and Graph Grammars", *Decision Support Systems*, Vol. 10, pgs. 255-275.

[Jones 94] C.V. Jones (1994) - "Visualization and Optimization", *ORSA Journal on Computing*, Vol. 6(3), pgs. 221-257.

[Jones 95] C.V. Jones (1995) - "Developments in Graph-Based Modeling for Decision Support", *Decision Support Systems*, Vol. 13, pgs. 61-74.

[Jones & D'Souza 92] C.V. Jones & K. D'Souza (1992) - "Graph Grammars for Minimum Cost Network Flow Modeling", Technical Report, Faculty of Business Administration, Simon Fraser University, Burnaby, British Columbia, Canada.

[Kanellakis & Papadimitriou 80] P.C. Kanellakis & C.H. Papadimitriou (1980) - "Local Search for the Asymmetric Traveling Salesman Problem" *Operations Research* vol. 28 pgs. 1086-1099.

[Karp 72] R.M. Karp (1972) - "Reducibility Among Combinatorial Problems" - *Complexity of Computer Computations*, R.E. Miller & J.W. Thatcher (Eds), Plenum, New York, pgs. 85-104

[Karp 76] R.M. Karp (1976) - "A Patching Algorithm for the Non-Symmetric Traveling Salesman Problem." *SIAM Journal of Computing* vol. 8 pgs. 561-573.

[Keen & Scott-Morton 78] P.G.W. Keen & M.S. Scott-Morton (1978) - *Decision Support Systems, An Organizacional Perspective*, Reading, MA; Addison-Wesley.

[Kesel'Man 87] D.Y. Kesel'Man (1987) - "Covering the Edges of a Graph by Circuits" - *Kibernetica*, vol. 3 pgs. 16-22

[Koskosides et al 92] Y.A. Koskosidis, W.B. Powell & M. Solomon (1992) - "An Optimization Based Heuristic for Vehicle Routing and Scheduling with Soft Time Window Constraints" - *Transportation Science*, Vol. 26(2) pgs. 69-85.

[Koskosides & Powell 92] Y.A. Koskosidis & W.B. Powell (1992) - "Clustering Algorithms for Consolidation of Customer into Vehicle Shipments" - *Transportation Science*, Vol. 26B(5) pgs. 365-379.

[Knuth 93] D. E. Knuth (1993) - *The Stanford GraphBase, A platform for combinatorial computing* - Stanford University, Addison-Wesley.

[Krishnan 90] R. Krishnan (1990) - "A Logic Modeling Language for Model Construction", *Decision Support Systems*, Vol. 6, pgs. 123-152.

[Kwan Mei-Ko 62] Kwan, Mei-Ko (1962) - "Graphic Programming Using Odd or Even Points" - *Chinese Mathematics*, vol. 1 pgs. 273-277

[Langevin, Soumis & Derosiers 89] A. Langevin, F. Soumis & J. Desrosiers (1989) - "Classification of Travelling Salesman Problem Formulations" - *Les Cahiers du GERAD*, G-88-36 HEC/Montreal-Canada

[Laporte 86] G. Laporte (1986) - "Generalized subtour elimination constraints and connectivity constraints" - *Journal of Operations Research Society*, Vol. 37 pgs. 509-514

[Laporte 92] G. Laporte (1992) - "The Traveling Salesman Problem: An Exact and Approximate Algoritms", *EUJOR* Vol. 59 pgs. 231-247.

[Laporte 92] G. Laporte (1992) - "The Vehicle Routing Problem: An overview of exact an approximate algorithms" - *EJOR*, vol 59, pgs. 345-358.

[Laporte & Nobert 87] G. Laporte & Y. Nobert (1987) - "Exact Algorithms for the Vehicle Routing Problem", *Annals of Discret Mathematics*, Vol. 31 pgs. 147-184

[Laporte et al 84] G. Laporte, Y. Nobert, D. Arpin (1984) - "Optimal Solutions to Capacitated Multi-Depot Vehicle Routing", *Congressus Memorantium*, Vol. 44 pgs. 283-292

[Laporte et al 85] G. Laporte, Y. Nobert, M. Desrochers (1985) - "Optimal Routing under Capacity and Distance Restrictions", *Operations Research*, Vol. 33 pgs. 1050-1073

[Laporte et al 86] G. Laporte, Mercure & Y. Nobert (1986) - "An Exact Algorithm for the Asymmetrical Capacitated Vehicle Routing Problem", *Networks*, Vol. 16 pgs. 33-46

[Laporte & Martelo 90] G. Laporte & S. Martello (1990) - "The Selective Travelling Salesman Problem" - *Discrete Applied Mathematics*, Vol. 26 pgs. 193-207

[Larson & Odoni 81] R.C. Larson & Amadeo R. Odoni (1981) - *Urban Operations Research* - Prentice-Hall, New Jersey.

[Lawler et al 85] E.L. Lawler, J. K. Lenstra, A.H.G. Rinnooy Kan, D. B. Shmos (1985) - *The Traveling Salesman Problem, A Tour of Combinatorial Optimization*, John Wiley & Sons, UK.

[Lee 93] R.M. Lee (1993) - "Direct Manipulation of Graph-Based Decision Models", *Decision Support Systems*, Vol. 9, pgs. 293-411.

[Lenstra & Rinooy Kan 81] J. Lenstra & A. Rinooy Kan (1981) - "Complexity of Vehicle Routing and Scheduling Problems" - *Networks*, Vol. 11 pgs. 221-227

[Levy & Bodin 88] L. Levy & L.D. Bodin (1988) - "Scheduling the Postal Carriers for the United States Postal Service: An Application of Arc Partitioning and Routing" - in *Vehicle Routing; Methods and Studies* B. Golden, A. Assad (Eds). North Holland, Amsterdam, pgs. 359-394.

[Levy & Bodin 89] L. Levy & L.D. Bodin (1989) - "The Arc Oriented Location Routing Problem" - *INFOR*, vol. 27 pgs. 74-94.

[Lin 65] S. Lin (1965) - "Computer Solutions for the Traveling Salesman Problem", *Bell Systems Computer Journal* Vol. 44 pgs. 2245-2269.

[Lin & Kerningham 73] S. Lin & B.W. Kerningham (1973) - "An Effective Heuristic Algorithm for the Traveling-Salesman Problem", *Operations Research* Vol. 21 pgs. 498-516.

[Lucena 86] A. Lucena (1986) - *Exact Solution Approaches for the Vehicle Routing Problem*, Ph.D Thesis, Imperial College of Science and Technology, University of London, England 205 pgs.

[Maculan et al 92] N. Maculan, M. Negreiros Gomes & L.S. Ochi (1992) – "A New Heuristic to Solve the Travelling Purchaser Problem", *Presented at the EURO XII/TIMS XXXI* - Helsinki, Finland.

[Maculan 96] N. Maculan (1996) – "Uma Formulação para o Problema de Compras", *Notas de Aula - COPPE/Sistemas*, Agosto

[Mak & Morton 93] K-T. Mak & A.J. Morton (1993) - " A Modified Lin-Kernigham Traveling-Salesman Heuristic.", *Operations Research Letters* vol. 13 pgs. 127-132.

[Martello & Toth 90] S. Martello & P. Toth (1990) - *Knapsack Problems, Algorithms and Computer Implementations*, John Willey & Sons.

[Male et al 77] J.W. Male, J.C. Lieberman & C. Orloff (1977) - "Improvement of Orloff's General Routing Problem" - *Networks*, vol. 7 pgs. 89-92

[Malek et al 89] M. Malek, A. Mourad & M. Pandria (1989) - "Topological Testing" - *Preceedings of the IEEE 1989 International Test Conference*, paper 4.4. pgs. 103-110

[McBridge 82] R. McBridge (1982) - "Controlling Left and U-turns in the Routing of Refuse Collection Vehicles", *Computers & Op. Res.*, vol. 9(2) pgs. 145-152

[Miller 95] D. Miller (1995) - "A Matching Based Exact Algorithm for Capacitated Vehicle Routing Problems", - *ORSA Journal on Computing*, vol. 7(1), pgs. 1-9.

[Minieka 79] E. Minieka (1979) - "The Chinese Postman Problem for Mixed Networks" - *Management Science*, vol. 23(7) pgs. 643-658.

[Miller & Pekny 91] D.L. Miller & J. F. Pekny (1991) - "Exact Solution of Large Asymmetric Traveling Salesman Problems", *Science*, vol. 251 pgs. 754-761

[Michelon et al 94] P. Michelon, M. Cruz & V. Gascon (1994) - "Using the Tabu Search Method for the Distribution of Supplies in a Hospital" - *Annals of Operations Research*, Vol. 50 pgs. 427-437

[Min et al 92] H. Min, J. Current & D. Schilling (1992) - "The Multiple Depot Vehicle Routing Problem with Backhauling" - *J. of Business Logistics*, Vol. 13(1) pgs. 259-288

[Mole & Jameson 76] R.H. Mole & S.R. Jameson (1976) - "A Sequential Route-Building Algorithm Employing Generalised Savings Criterion" - *Op. Res. Quarterly*, Vol. 27 pgs. 503-511

[Murphy et al 92] F.H. Murphy, E.A. Stohr & Asthana (1992) - "Representation Schems for Linear Programming Models", *Management Science*, Vol. 38(7), pgs. 964-991.

[Mulvey & Beck 84] J.M. Mulvey & M.P. Beck (1984) - "Solving Capacitated Clustering Problems", *EJOR*, Vol. 18, pgs. 339-348.

[Nagl 76] M. Nagl (1976) - "Formal Languages of Labeled Graphs", *Computing*, Vol. 16, pgs. 113-137.

[Nagl 79] M. Nagl (1979) - "A Tutorial Bibliographical Survey on Graph-Grammars", *Graph-Grammars and Their Application to Computer Science and Biology*, Claus, Ehrig & Rosenberg (Eds) Vol. Springer-Verlag, Berlin, pgs. 70-126.

[Negreiros 90] M.J. Negreiros Gomes (1990) - *O Problema de Planejamento e Percurso de Veículos na Coleta do Lixo Urbano Domiciliar* - MSc. Dissertation *UFRJ- COPPE/Sistemas*,

[Negreiros 93] M. Negreiros Gomes (1993) - "Melhorias no Algoritmo de Clarke & Write" - *Anais do XXV SBPO* - pgs. 467-472, SOBRAPO-Campinas/SP.

[Negreiros & Maculan 92a] M.J. Negreiros Gomes, & N. Maculan (1992) - "MARCOLIX, A Capacitated Arc Routing Heuristic Algorithm Applied to Garbage Collection" - *Relatório Técnico, UFRJ-COPPE/Sistemas*, ES-254/92.

[Negreiros & Maculan 92b] M.J. Negreiros Gomes, & N. Maculan (1992) - "Minimizing the Fleet Size in Service in Large Areas Considering Fixed Frequency of Work." - *Relatório Técnico, UFRJ-COPPE/Sistemas*, ES-257/92.

[Negreiros & Maculan 92] M.J. Negreiros Gomes & N. Maculan (1992) - "The Problem of Planning a Fleet of Vehicles in the Garbage Collection." - *Relatório Técnico, UFRJ-COPPE/Sistemas*, ES-258/92.

[Negreiros et al 92] M.J. Negreiros Gomes, F.J. Negreiros Gomes & N. Maculan (1992) - "A Reserva Mínima Estocástica de Máquinas" - *Anais do XXIV SBPO, Salvador/BA*, pgs. 119-123.

[Negreiros et al 92a] Negreiros Gomes, N. Maculan & D.F. Claiman - "Optimizing Routes in Household Refuse Collection in Rio de Janeiro (A Field Application Using S i s R o t" - *Relatório Técnico, UFRJ-COPPE/Sistemas*, ES-255/92.

[Negreiros et al 92b] M.J. Negreiros Gomes, F.J. Negreiros Gomes, N. Maculan & N. Vasconcelos (1992) - "The Minimal Fleet and Stock of Vehicles in a Large Systematic Service Areas" - *Relatório Técnico, UFRJ-COPPE/Sistemas*, ES-256/92.

[Negreiros et al 92] M. Negreiros Gomes, F. Negreiros Gomes & F.A. Zampirolli (1992) - "SisGRAFO, A Graphical Interactive System to Solve Graph Optimization Problems" - *Presented at the VI CLAIO* - Mexico City, Mexico.

[Negreiros et al 94] M. Negreiros Gomes, N. Maculan & L.S. Ochi (1994) - "O Problema de Compras" - *Pesquisa Operacional* - vol 14(2), pgs. 73-86 (publicado em 1996).

[Negreiros & Maculan 94] M. Negreiros Gomes & N. Maculan (1994) - "Problema de Agrupamento Capacitado - Refinamento para Heurística Primal" - *Anais do XXVI SBPO*, pgs. 533-538 Ed. SOBRAPO.

[Negreiros & Negreiros 94] M. Negreiros Gomes & F. Negreiros Gomes (1994) - "Resolvendo Problemas de Distribuição de Derivados de Petróleo com o SisGRAFO" - *Anais do XXVI SBPO*, pgs. 527-532 Ed. SOBRAPO.

[Negreiros 96] M. Negreiros Gomes (1996) - *Projeto de Engenharia para Localização de uma Estação de Transferência de Lixo na Cidade de Fortaleza*, Relatório Técnico, EMLURB, Fortaleza/Ce.

[Nelson et al 85] M. Nelson, K. Nygard, J. Griffin & W. Shreve (1985) - "Implementation Technics for the Routing Problem" - *Comp. & Ops. Res.*, Vol. 12 pgs. 273-283.

[Nobert & Picard 91] Y. Nobert & J-C. Picard (1991) - "An Optimal Algorithm for the Mixed Chinese Postman Problem" - *CRT*, no. 799, Montreal.

[Nygard et al 90] K. Nygard, P. Juell & N. Kabada (1990) - "Neural Networks for Selecting Vehicle Routing Heuristics" - *ORSA Journal on Computing*, Vol. 2 pgs. 353-364.

[Nurmi 91] K. Nurmi (1991) - "Travelling Salesman Problem Tools for Microcomputers" - *Computers & Op. Research*, vol. 18(8) pgs. 741-749.

[Ochi 89] L. Ochi (1989) - "Contribuições à Solução de Problemas de Percurso e Sequenciamento de Veículos", - *DSc. Thesis*, Universidade Federal do Rio de Janeiro - COPPE/Eng. Sistemas.

[Ochi 94] L.S. Ochi (1994) - "Conhecimento heurístico: Aplicações em problemas de Otimização." - XIV Congresso da SBC, XIII Jornada de Atualição em Informática, Caxambu/MG.

[Ochi & Maculan 89] L. Ochi & N. Maculan (1989) - "Routing and Scheduling of Tanker Fleet with Time Windows", - *ALIO Workshop on Pratical Combinatorial Optimization*, Rio de Janeiro/RJ.

[Ochi & Maculan 91] L.S. Ochi & N. Maculan (1991) – "The Travelling Purchaser Problem" - *Presented at the XXX TIMS* (July) – Rio de Janeiro, Brazil.

[Ochi & Negreiros 92] L. Ochi & M. Negreiros (1992) - "Sistemas Computacionais de Apoio a Problemas de Roteamento de uma Frota de Veículos", - *Relatório Técnico - COPPE/Eng. Sistemas* ES-259/92, Universidade Federal do Rio de Janeiro.

[Ochi et al 92] L.S. Ochi, N. Maculan & M. Negreiros Gomes, (1992) – "The Travelling Purchaser Problem with Time Windows" - *Presented at the VI CLAIO* – Mexico City, Mexico.

[Or 76] I. Or (1976) - *Traveling Salesman-type Combinatorial Problems and their Relation to the Logistics of Regional Blood Banking.* - Ph.D. Dissertation, Northwestern University, Evanston, IL

[Orloff 74] C. Orloff (1974) - "A Fundamental Problem on Vehicle Routing" - *Networks*, Vol. 4 pgs. 34-64

[Orloff 76] C. Orloff (1976) - "Routing-Constrained Fleet Scheduling" - *Transp. Science*, Vol. 10 pgs. 149-168

[Osman 93] I.H. Osman (1993) - "Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem", - *Annals of Operations Research*, Vol. 41 pgs. 421-451

[Padberg & Grötschel 85] M.W. Padberg & M. Grötschel (1985) - "Polyhedral Computations" - *Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Lawler, Lenstra, Rinnooy Kan & Shimoys (Eds.) pgs. 307-360. Wiley, Chichester.

[Padberg & Rao 82] M.W. Padberg & M.R. Rao (1982) - "Odd Minimum Cut-Sets and b-Matchings" - *Math. Oprs. Res.*, Vol. 7 pgs. 67-80

[Papadimitriou & Steiglitz 82] C.H. Papadimitriou & K. Steiglitz (1982) - *Combinatorial Optimization: Algoritms and Complexity*, Prentice Hall, Englewood Cliffs, NJ.

[Paessens 88] H. Paessens (1988) - "The Savings Algorithm for the Vehicle Routing Problem", - *EJOR*, Vol. 34 pgs. 336-344

[Pearn 84] W.-L Pearn (1984) - *The Capacitated Chinese Postman Problem* - Ph.D. Thesis, University of Maryland.

[Pearn 88] W.-L Pearn (1988) - "New Lower Bounds for the Capacitated Arc Routing Problem" - *Networks*, vol 18 pgs. 181-191

[Pearn 89] W.-L Pearn (1989) - "Approximate Solutions for the Capacitated Arc Routing Problem" - *Computers & Operations Research*, vol 16 pgs. 589-600

[Pearn 91] W.-L Pearn (1991) - "Augment-Insert Algorithms for the Capacitated Arc Routing Problem" - *Computers & Operations Research*, vol 18 pgs. 189-198

[Pearn et al 87] W.-L Pearn (1987) - "Transforming Arc Routing into Node Routing Problems" - *Computers & Operations Research*, vol 14 pgs. 285-288

[Pedder 90] S. Pedder (1990) - *An Integer Model for Scheduling and Routeing a Groceries Distribution Vehicle Fleet*, - MSc. Dissertation, University of Aukland, New Zealand/USA.

[Platzman & Bartholdi 89] L.K. Platzman & J.J. Bartholdi (1989) - "Spacefilling Curves and the Planar Travelling Salesman Problem", *Journal of ACM*, vol. 36 pgs. 719-737

[Potvin & Rousseau 93] J.Y. Ptovin (1993) - "A Parallel Route Building Algorithm for the Vehicle Routing and Scheduling Problem with Time Windows", - *EUJOR*, Vol. 66(3) pgs. 331-340

[Potvin et al 94] J-Y. Potvin, G. Lapalme & J-M. Rousseau (1994) - "A Microcomputer Assistant for the Development of Vehicle Routing and Scheduling Heuristics" - *Decision Support Systems*, vol 12 pgs. 41-56

[Psaraftis 83] H. Psaraftis (1983) - "An Exact Algorithm for the Single Vehicle Many-to-Many Dial-a-Ride Problem with Time Windows", - *Transportation Science*, vol. 17 pgs. 351-357.

[Pureza & França 91] V.M. Pureza & P.M. França (1991) - "Vehicle Routing Problems via Tabu Search Metaheuristic", - *Publication CRT-747*, Centre de Recherche sur les Transports, Université de Montréal.

[Ramesh 78] R. Ramesh (1978) - Unpublished Monography - refered at the Golden et al (1981)

[Ramesh 91] R. Ramesh & K. M. Brown (1991) - "An Efficiente Four-Fase Heuristic for the Generalized Orienteering Problem" - *Computers & Operations Research*, vol. 18(2), pgs. 151-165.

[Ramesh et al 92] R. Ramesh, Y-S Yoon & M.H. Karwan (1992) - "An Optimal Algorithm for the Orienteering Problem" - *ORSA Journal on Computing*, vol. 4(2), pgs. 155-165.

[Rego 96] C. Rego (1996) - "A Subpath Ejection Method for the Vehicle Routing Problem", - *Relatório Técnico*, Universidade Portucalense.

[Rego & Roucariol 96] C. Rego & C. Roucariol (1996) - "A Parallel Tabu Search Algorithm Using Ejection Chains for the Vehicle Routing Problem", - *Relatório Técnico*, Universidade Portucalense.

[Ribeiro & Soumis 94] C. Ribeiro & F. Soumis (1994) - "A Column Generation Approach to the Multiple-Depot Vehicle Scheduling Problem", - *Op. Research*, vol. 42(1), pgs. 41-52.

[Rochat & Taillard 95] Y. Rochat & É.D. Taillard (1995) - "Probabilistic Diversification and Intensification in Local Search for Vehicle Routing", - *Journal of Heuristics*, vol. 1, pgs. 147-167.

[Ronen 92] D. Ronen (1992) - "Allocation of Trips to Trucks Operating from a Single Terminal", - *Comp. Op. Research*, vol. 19(5), pgs. 445-451.

[Ronen 95] D. Ronen (1995) - "Dispatching Petroleum Products", - *Op. Research*, vol. 43(3), pgs. 379-387.

[Rosenkrants et al 77] D.J. Rosenkrantz, R.E. Stearns & P.M. Lewis (1977) - "An Analysis of Several Heuristics for the Travelling Salesman Problem", *SIAM Journal on Computing*, vol. 6 pgs. 563-581

[Roy & Rousseau 89] S. Roy & J.M. Rousseau (1989) - "The Capacitated Canadian Postman Problem" - *IFORS*, Vol. 27(1) pgs. 58-73

[Russel 77] R. Russell (1977) - "An Effective Heuristic for M-Tour Traveling Salesman Problem with Some Side Constraints", *Op. Research*, vol. 25(3) pgs. 517-524.

[Russel & Challinor 88] R. Russell & P. Challinor (1988) - "Effective Methods for Petroleum Tank Truck Dispatching" - *Comp. Op. Research*, vol. 15(4) pgs. 323-331.

[Russel & Igo 79] R. Russell & W. Igo (1979) - "An Assignment Routing Problem" - *Networks*, vol. 9 pgs. 1-17.

[Ryan et al 90] D. Ryan, C. Hjorring & F. Glover (1990) - "Extensions of the Petal Method for Vehicle Routing" - *Technical Report*, University Auckland.

[Salhi et al 92] S. Salhi, M. Sari, D. Saidi & NAC. Touati (1992) - "Adaptation of Some Vehicle Fleet Mix Heuristics" - *OMEGA*, Vol. 20(5/6) pgs. 653-660.

[Sahni & Gonzalez 76] S.K. Sahni & T. Gonzalez (1976) - "P-Complete Approximations Problems", *Journal of the ACM*, vol. 23 pgs. 555-565.

[Salvesbergh 84] M.W.P. Salvesbergh (1984) - "Local Search in Routing Problems with Time Windows", - *Report OS-R8409*, Centre for Mathematics and Computer Science, Amsterdam.

[Salvesbergh 90] M.W.P. Salvesbergh (1990) - "An Efficient Implementation of Local Search Algorithms for Constrained Routing Problems", - *EJOR*, vol. 47, pgs. 75-85.

[Salvesbergh 92] M.W.P. Salvesbergh (1992) - "The Vehicle Routing Problem with Time Windows: Minimizing Route Duration", - *ORSA Journal on Computing*, vol. 4, pgs. 146-154.

[Semet & Taillard 93] F. Semet & É.D. Taillard (1993) - "Solving Real-Life Vehicle Routing Problems Efficiently Using Tabu Search", - *Annals of Operations Research*, vol. 41 pgs. 469-488.

[Sexton & Bodin 85a] T. Sexton & L. Bodin (1985a) - "Optimizing Single Vehicle Many-to-Many Operations with Desired Delivery Times: I. Scheduling" - *Transportation Science*, vol. 19 pgs. 378-410

[Sexton & Bodin 85b] T. Sexton & L. Bodin (1985b) - "Optimizing Single Vehicle Many-to-Many Operations with Disired Delivery Times: I. Scheduling" - *Transportation Science*, vol. 19 pgs. 411-435

[Sherafat 88] H. Sherafat (1988) - "Uma Solução para o Problema do Carteiro Chinês Misto", *Anais do IV CLAIO - XXI SBPO*, pgs. 157-170, Rio de Janeiro

[Simon 77] H. Simon (1977) - *The New Science of Management Decision*, Englewood Cliffs, NJ. Prentice Hall.

[Skitt & Levary 85] R.A. Skitt & R.R. Levary (1985) - "Vehicle Routing Via Column Generation" - *EJOR*, Vol. 21 pgs. 65-76

[Solomon 87] M. Solomon (1987) - "Algorithms for Vehicle Routing and Scheduling Problems with Time Windows Constraints", - *Operations Research*, vol. 35 pgs. 254-265.

[Solomon et 88] M. Solomon, E. Baker & J. Schaffer (1988) - "Vehicle Routing and Scheduling Problems with Time Window Constraints: Efficient Implementation of Solution Improvement Procedures", - *Vehicle Routing Methods and Studies*, pgs. 85-105 Ed. Golden & Assad, North Holland.

[Solomon & Desrosiers 88] M. Solomon & J. Desrosiers (1988) - "Time Window Constrained Routing and Scheduling Problems", - *Transportation Science*, vol. 22 pgs. 1-13.

[Stern & Dror 79] H. Stern & M. Dror (1979) - "Routing Electric Meter Readers", *Computers & Op. Res.*, Vol. 6 pgs. 209-223

[Stewart & Golden 84] W.R. Stewart & B.L Golden (1984) - "A Lagrangean Relaxation Heuristic for Vehicle Routing", - *EJOR*, vol. 15 pgs. 84-88.

[Swersey & Bellard 84] A. Swersey & W. Bellard (1984) - "Scheduling School Buses" - *Management Science*, vol. 30(7) pgs. 161-174.

[Syslo et al 83] M. Syslo, N. Deo & J. Kowalik, J. (1983) - *Discrete Optimization Algorithms with Pascal Programs*, Prentice Hall,Ed.

[Swan 93] T. Swan (1993) - *Programando em Pascal 7.0 para Windows*, Rio de Janeiro, Ed. Berkeley.

[Tan & Beasley 84] C.C.R. Tan & J. Beasley (1984) - "A Heuristic Algorithm for the Period Vehicle Routing Problem", *OMEGA*, Vol. 12, pgs. 497-504.

[Taillard 93] É.D. Taillard (1993) - "Parallel Interative Search Methods for Vehicle Routing Problems", - *Networks*, vol. 23, pgs. 661-673.

[Trudeau & Dror 92] P. Trudeau & M. Dror (1992) - "Stochastic Inventory Routing: Route Design with Stockouts and Route Failures", *Transportation Science*, Vol. 26(3), pgs. 171-184.

[Turban 95] E. Turban (1995) - *Decision Support and Expert Systems* - 4th. Ed. Pentice-Hall International, North Holland.

[Ulosoy 85] G. Ulosoy (1985) - "The Fleet Size and Mix Problem for Capacitated Arc Routing" - *European Journal of Op. Res.*, Vol 22 pgs 329-337

[Volgenant & Jonker 82] T. Volgenant & R. Jonker (1982) - "A Branch and Bound Algorithm for the Symmetric Travelling Salesman Problem based on the 1-tree Relaxation", *EJOR*, vol. 9 pgs. 83-89.

[Xu & Kelly 96] J. Xu & J.P. Kelly (1996) - "A Network Flow-Based Tabu Search Heuristic for the Vehicle Routing Problem", - *Relatório Técnico*, Graduate School of Business, University of Colorado at Boulder.

[Yaxiong & Yongchang 88] Yaxiong Lin & Z. Yongchang (1988) - "A New Algorithm for the Direct Chinese Postman Problem" - *Comp. & Oper. Res.*, Vol. 15 pgs. 577-584

[Yellow 70] P.C. Yellow (1970) - "A Computacional Modification to the Savings Method of Vehicle Scheduling", - *Op. Res. Quarterly*, vol. 21(2) pgs. 281-283.

[Wark & Holt 94] P. Wark & J. Holt (1994) - "A Repeated Matching Heuristic for the Vehicle Routeing Problem", - *JORS*, vol. 45(10) pgs. 1156-1167.

[Willard 89] J.A. Willard (1989) - "Vehicle Routing Using r-Optimal Tabu Search", - *M.Sc. Dissertation*, The Management School, Imperial College, London.

[Win 87] Z. Win (1987) - *Contributions to Routing Problems* - Doctoral Dissertation, Universität Augsburg. pgs. 97-112

[Win 89] Z. Win (1989) - "On the Windy Postman Problem on Eulerian Graphs" - *Mathematical Programming*, Vol. 44 pgs. 97-112

[WHA 96] "Dengue and Dengue Haemorraghic Fever Control" - *WHA Home Page*, Kazem Behbehani.

[Weiss 92] M.A. Weiss (1992) - *Data Structures and Algorithm Analysis*, Benjamim/Cummings Pub., 1st Ed. California/USA

[Wren 71] A. Wren (1971) - *Computers in Transport Planning and Operation*, Ian Allan (Ed), London.

[Wren & Holliday 72] A. Wren & A. Holliday (1972) - "Computer Scheduling of Vehicles from One or More Depots to a Number of Delivery Points", - *Op. Res. Quarterly*, vol. 23 pgs. 333-344.

[Wolfberg 69] M.S. Wolfberg (1969) - *An Interactive Graph Theory System*, Ph.D. Dissertation, University of Pennsylvania, Philadelphia, PA

[Wong 80] R. T. Wong (1980) - "Integer Programming Formulations of the Travelling Salesman Problem", *Proceedings of the IEEE International Conference of Circuits and Computers*, pgs. 149-159.

[Wunderlich et al 92] J. Wunderlich, M. Collette, L. Levy & L. Bodin (1992) - "Scheduling Meter Readers for Southern California Gas Company", *Interfaces*, Vol. 22(3) pgs. 22-30