UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

Algoritmos Heurísticos de Cobertura de Arcos

Tese de Doutorado

Hassan Sherafat

Autor

Sérgio Fernando Mayerle, Dr.

Orientador

Florianópolis

Santa Catarina – Brasil

Março de 2004

Folha de Aprovação

Algoritmos Heurísticos de Cobertura de Arcos

Tese defendida e aprovada em 25 de Março de 2004

Autor: Hassan Sherafat

Banca Examinadora:

Prof. Sérgio Fernando Mayerle, Dr. – Orientador

Profa. Maria Cristina Sinay, Dra. – Examinadora Externa

Profa. Mirian Buss Gonçalves, Dra. – Examinadora

Prof. Celso Carnieri, Dr. – Examinador Externo

Prof. João Carlos de Souza, Dr. – Examinador

Prof. João Neiva de Figueiredo, Dr. – Examinador

Prof. Edson Pacheco Paladini, Dr. – Coord. Pós-Graduação

Dedico este trabalho à memória de meus pais,
Tahereh Bazzazan Sherafat e Nasrul'láh Sherafat

Agradecimentos

Considerando que nenhum trabalho é uma realização individual, quero registrar meus agradecimentos a todas as pessoas e Instituições que direto, ou indiretamente cotribuiram para a concretização deste trabalho, em especial:

Ao meu orientador, Prof. Dr. Sérgio Fernando Mayerle, pela sua dedicação e uma orientação precisa durante todo o período do curso; mas não apenas por isso, ainda mais, pelo exemplo que me deixou como um professor com elevadas qualidades intelectuais e morais, humildemente canalizadas ao serviço da Universidade;

Aos integrantes da Banca Examinadora, professores João Neiva de Figueiredo, Mirian Bus Gonçalves, João Carlos de Souza, Maria Cristina Sinay, e Celso Carnieri, pelas valiosas sugestões que contribuiram para a apresentação deste trabalho na sua forma final;

À Universidade Federal de Sergipe que me proporcionou a oportunidade de aprimoramento acadêmico e, em especial, aos colegas do Departamento de Matemática que solidariamente possibilitaram a mesma;

À CAPES, pelo apoio financeiro;

À COMCAP, pelo pronto atendimento e fornecimento de dados utilizados na pesquisa, em particular aos Engenheiros Hélio, Edmar, e Wilson.

****

Devo uma gratidão toda especial à minha querida esposa e eterna companheira Felora Daliri Sherafat com seu apoiou em todos os momentos deste trabalho e suas palavras de estímulo e encorajamento;

E aos meus queridos filhos, Vahid e Tahereh, que com seu amor e carinho característicos deram significado aos meus esforços.

Resumo

Nos problemas de roteamento o objetivo é determinar um circuito de custo mínimo que cobre um dado conjunto de arcos ou nós de um grafo, sujeito a algumas restrições. Existem duas classes bem conhecidas de tais problemas, denominadas como o Problema de Caixeiro Viajante (PCV), e o Problema do Carteiro Chinês (PCC). Com raras exceções, todos os problemas já formulados nessas duas classes são NP-completos. Portanto, para os problemas de maior porte existem apenas soluções aproximadas.

Nessa Tese foi considerado o problema de determinar um circuito de custo mínimo que cobre um dado subconjunto de arcos, de arestas e de nós de um grafo misto, sujeito a algumas restrições nos vértices (restrições que proíbem conversões indesejáveis nos cruzamentos de malhas urbanas). Obviamente, o PCV, PCC e a maior parte de suas variações, como: o Problema do Carteiro Chinês Misto e o Problema do Carteiro Rural são casos particulares deste problema geral. A solução proposta é baseada numa transformação polinomial do grafo que possibilita a solução do problema resultante como um PCV padrão. Resultados computacionais confirmam a eficiência do método na obtenção de soluções próxima a ótimas para problemas razoavelmente grandes.

Palavras-chave: Roteamento de Arcos, Problema do Carteiro Chinês Misto, PCCM, Problema do Carteiro Rural, PCR, Problema Geral de Roteamento, PGR, Restrições de Conversão, Coleta de Lixo.

Abstract

In routing problems, the aim is to determine a least cost circuit covering a specified set of arcs or nodes of a graph, subject to some constraints. There are two well-known classes of such problems, called as Traveling Salesman Problem (TSP) and Chinese Postman Problem (CPP). With rare exceptions, all problems already formulated in these two classes are NP-complete; therefore only approximate solutions are available for reasonable-sized problems.

In this paper, we consider the problem of determining a least cost circuit which covers a given subset of arcs, edges and nodes of a mixed graph, subject to some node restrictions (restrictions that avoid bad turns of vehicles in real-life street networks). Obviously, the TSP, CPP and the major part of its variations, such as the Mixed Chinese Postman Problem and the Rural Postman Problem are particular cases of this general problem. Our solution is based on an efficient graph transformation that makes it possible to solve the resulting problem as a standard TSP. Computational results confirm the efficiency of the method for solving relatively large problems with good solution quality.

Key-words: Arc Routing Problem, Mixed Chinese Postman Problem, CPP, Rural Postman Problem, RPP, General Routing Problem, GRP, Turn Penalty, Solid Waste Collection.

Algoritmos Heurísticos de Cobertura de Arcos

ÍNDICE

Glossário das Abreviaturas Usadas para os Problemas de Roteamento

PCC	Problema de Carteiro Chinês
PCCM	Problema de Carteiro Chinês Misto
PCCO	Problema de Carteiro Chinês Orientado
PCCV	Problema de Carteiro com Vento
PCR	Problema de Carteiro Rural
PCRA	Problema de Carteiro Rural Agrupado
PCRM	Problema de Carteiro Rural Misto
PCRMCP	Problema de Carteiro Rural Misto com Conversões Penalizadas
PCRO	Problema de Carteiro Rural Orientado
PCRV	Problema de Carteiro Rural com Vento
PCV	Problema de Caixeiro Viajante
PCVA	Problema de Caixeiro Viajante Assimétrico
PCVG	Problema de Caixeiro Viajante Generalizado
PE	Problema da Empilhadeira
PGR	Problema Geral de Roteamento
PGRCP	Problema Geral de Roteamento com Conversões Penalizadas
PRA	Problema de Roteamento de Arcos
PRAC	Problema de Roteamento de Arcos Capacitado
PRN	Problema de Roteamento de Nós
PRV	Problema de Roteamento de Veículos
PRVMI	Problema de Roteamento de Veículos com Múltiplas Instalações

I - Introdução

1.1 Problemas Básicos de Logística

A distribuição de bens e serviços a um conjunto de consumidores, partindo de alguns pontos de abastecimento, é um dos problemas que mais cresce de importância e relevância, não apenas pelo crescimento vegetativo da economia mundial, mas principalmente devido aos paradigmas do comércio moderno. De um lado, há uma tendência de migração dos mercados produtores conforme a vocação regional. Isso pode significar um distanciamento maior destes produtores dos mercados de consumo. Porém, por outro lado, o comércio eletrônico disponibiliza estes produtos e serviços a todo consumidor potencial, sem limite geográfico. Cada vez menos ele precisa sair de seu lugar na busca de produtos e serviços. Paradoxalmente, na mesma medida que os centros produtores se distanciaram do mercado consumidor, os produtos em si chegaram muito mais próximos ao seu destino final. Essa crescente distância é coberta por um complexo sistema logístico em funcionamento em todo o planeta.

Para se ter uma idéia da grandeza do sistema logístico, abaixo estão relatados alguns dados de natureza conjuntural. Segundo *Bureau of Transportation Statistics – BTS* [BTS03], o comprimento das vias rodoviárias públicas dos EUA (estradas, malhas urbanas, vias rurais) foi de 6,34 milhões de km no ano de 2000, o que equivale a 158 voltas à Terra. As frotas que rodaram nestas vias foram formadas por 7,85 milhões de veículos de carga (excluidos os veículos de carga que não faziam parte de frotas), percorrendo uma distância total de 331 bilhões de km no período. A movimentação rodoviária de cargas nesse mesmo ano foi de aproximadamente 4 trilhões de ton×km. Em 1999, as despesas com transporte de cargas nesse país totalizaram US$ 562 bilhões, dos quais US$ 457 bilhões foram com transporte rodoviário de cargas. O setor de transportes, incluindo outros modais, foi responsável por 3,2% do PIB americano em 2000.

No Brasil, segundo os dados do *IBGE* [IBG03], em 1999 a participação do setor de transportes no PIB brasileiro foi de 4,4%, resultado de um volume de produção de R$ 15,8 bilhões. Segundo os levantamentos da *GEIPOT* [Gei00], em 1999 as rodovias brasileiras tiveram um comprimento total de 1,72 milhões de quilômetros, rodando nelas 1,84 milhões de veículos de carga, movimentando 447 bilhões de ton×km de cargas.

Como um exemplo específico, pode ser mencionado o caso de coleta de lixo – um serviço amplamente oferecido pelas Prefeituras de quase todas as cidades brasileiras. Segundo dados levantados pelo *IBGE* [IBG03] no ano 2000, dos 5507 municípios do Brasil, 5471 dispunham deste serviço. Diariamente foram coletados 228.413,0 toneladas de lixo nesse período. Considerando que nas cidades de grande e médio porte os aterros sanitários se situam bem distantes dos centros urbanos, o serviço passa a ser realizado em algumas etapas que, em geral, compreendem: a coleta domiciliar, o tratamento numa estação apropriadamente localizada, e o transporte para o destino final. Isso mostra a complexidade, e conseqüentes gastos que envolvem este serviço.

O grande volume de recursos despendidos nas operações logísticas tem motivado os pesquisadores a desenvolverem modelos de otimização aplicados a diversas instâncias do problema. A depender da natureza do problema logístico, diversos problemas de otimização combinatorial já foram formulados e, muitos dos quais, resolvidos [Bod83]. Os problemas formulados podem ser classificados em três níveis:

- ***Estratégico***: a exemplo de decisões sobre a localização de fábricas, depósitos, filiais, estações de tratamento, e outras decisões que definem a estratégia de distribuição;
- ***Tático***: como decisões sobre o tamanho e tipo da frota;
- ***Operacional***: são as decisões que precisam ser tomadas a cada dia, com objetivo de aumentar a eficiência e reduzir o custo operacional do serviço, entre elas as escalas de mão-de-obra e de veículos.

Um dos principais problemas a ser resolvido neste item é o roteamento de veículos, o qual é o objeto de atenção desta tese.

Obviamente, as decisões de um nível hierárquico maior tem seu impacto no desempenho de um nível menor. Algumas abordagens procuram possibilitar o estudo de um modelo integrado, considerando simultaneamente a localização de instalações, dimensionamento da frota e a distribuição de produtos e / ou serviços [Geo74].

Numa grande parcela de problemas de distribuição, uma vez tomadas as decisões estratégicas e táticas, se torna altamente relevante a definição de um roteiro que minimize o custo de entrega do bem ou serviço. Na maioria dos casos, os custos operacionais associados aos veículos se configuram entre os mais significativos de todo o processo de distribuição, e em geral são sensíveis aos roteiros definidos para o cumprimento do

serviço. Nestes casos, pequenas percentagens de redução podem resultar em grandes economias no custo total da operação.

1.2 Problemas de Roteamento

Num problema de distribuição o objetivo final é atender a um conjunto de demanda distribuída numa rede. Uma rota é uma seqüência apropriada de locais a serem visitados para tal atendimento. Existem quatro entidades principais que influenciam uma rota, cada uma impondo um grupo de restrições ou condições que caracterizam o problema de roteamento, a saber:

- ***instalações***: são os locais de abastecimento; pode haver uma única, ou então múltiplas instalações, e cada instalação pode ter a sua própria capacidade;
- ***veículos***: a demanda pode ser suprida por um único veículo, ou por uma frota de *M* veículos, respeitando a capacidade de cada um;
- ***rede***: a rede pode ser orientada, não-orientada ou mista, e além disso podem existir outras restrições de transito impostas aos veículos;
- ***demanda***: a demanda pode estar localizada em pontos específicos da rede, ou distribuídas ao longo de trechos; ela pode ser uniforme, ou variável, e pode conter restrições temporais; pode, ainda, ser determinística, ou estocástica.

Considerando que uma rota começa numa instalação (fábrica, depósito, estação, etc.), percorre uma rede (uma malha viária), utilizando uma frota de veículos, com objetivo de visitar alguns pontos de demanda, a sua definição e o método de solução empregado variam enormemente conforme as restrições que cada uma destas entidades impõe ao problema.

De fato, as combinações destas restrições resultam numa variedade de formulações para os problemas de roteamento. Algumas formulações clássicas, obtidas a partir de simplificações nas restrições acima, e que formam os problemas básicos de roteamento, são relacionadas a seguir, nos quais a rede é representada por um grafo:

- ***Problema do Caixeiro Viajante – PCV***: requer a determinação de um circuito de custo mínimo que percorre todos os nós de um dado grafo;

- ***Problema do Carteiro Chinês – PCC***: consiste em determinar um circuito de custo mínimo que contém todos os links (arcos e / ou arestas) do grafo, pelo menos uma vez;
- ***Problema de Roteamento de Arcos Capacitado – PRAC***: é uma versão generalizada do PCC em que a cada link é associada uma demanda; portanto o problema consiste em gerar alguns circuitos, de custo total mínimo, a fim de atender toda a demanda e respeitar uma dada capacidade para cada circuito;
- ***Problema Geral de Roteamento – PGR***: é outra generalização estudada na literatura em que o PCC e o PCV aparecem como casos particulares; o objetivo é achar um circuito de custo mínimo que cobre um dado subconjunto de links e um dado subconjunto de nós do grafo;
- ***Problema de Roteamento de Veículos – PRV***: o problema é uma generalização do PCV em que um conjunto de *M* circuitos, com capacidades dadas, deve atender a demandas localizadas nos nós, todos partindo de uma mesma instalação, de modo que cada nó do grafo seja coberto por um único circuito e que o custo total seja mínimo;
- ***Problema de Roteamento de Veículos com Múltiplas Instalações – PRVMI***: é uma generalização do PRV em que há *p* instalações dispostas no grafo; um conjunto de *M* circuitos deve ser formado, cada um a partir de uma das instalações, respeitando a capacidades dos circuitos e das instalações, de modo que cada nó do grafo seja coberto por um único circuito e que o custo total seja mínimo.

1.3 Objetivos

Objetivo Geral: estudar algumas variações do Problema do Carteiro Chinês, e contribuir às suas soluções com algoritmos aproximados.

Objetivos Específicos: abordar o Problema do Carteiro Chinês no grafo Misto, bem como algumas generalizações que envolvem este caso, tais como: o Problema do Carteiro com Vento, o Problema do Carteiro Rural e o Problema Geral de Roteamento; considerar as restrições associadas a rede, inclusive aquelas relacionadas com as conversões nos vértices.

Não serão objetos desse estudo, os casos capacitados do problema de roteamento de arcos, nem consideradas as demais restrições dos problemas reais de roteamento, relacionadas com instalações, veículos e a demanda.

1.4 Relevância

A relevância desse estudo é devido a dois fatos: o primeiro é o fator econômico; os problemas de distribuição de bens e serviços, por razões anteriormente expostas, estão adquirindo cada vez mais importância e envolvendo maior volume de recursos. Os problemas de roteamento de arcos são o cerne e a etapa mais crítica na solução de muitos problemas de distribuição.

O segundo é a relevância do problema do ponto de vista teórico. Com raras exceções, os problemas de roteamento de arcos são NP-hard [Eis95.1]. Por conseguinte, na maioria dos casos soluções exatas são difíceis de serem obtidas e, mesmo quando existem, são ineficientes em termos computacionais. Soluções aproximadas, embora existam para vários casos, são poucas em número, e limitadas em termos da variedade de abordagens, comparadas com outras áreas de otimização combinatória.

A inserção de restrições da ordem de percurso nos vértices do grafo é um outro atrativo de relevância prática e teórica. Elas podem representar as conversões proibidas, ou indesejáveis, nos cruzamentos das vias urbanas. Embora estejam entre as mais importantes restrições dos problemas reais de roteamento, poucos trabalhos têm sido dedicados ao seu tratamento.

1.5 Aplicações

O Problema do Carteiro Chinês e suas variações têm uma grande gama de aplicações, na distribuição de serviços públicos, ou privados, na entrega ou coleta de mercadorias, e em vários outros problemas que podem ser formulados como tal. Alguns exemplos de aplicações que envolvem roteamento de arcos são as seguintes [Bod78]:

- coleta de lixo domiciliar;
- limpeza de ruas usando varredores mecânicos;
- remoção de neve das vias públicas;
- pulverização das vias públicas com sais que evitam a formação de gelo;
- serviço de entrega de cartas e encomendas de correios;
- inspeções periódicas em linhas elétricas, redes de gasodutos, ou oleodutos;
- serviços de transporte escolar;

- distribuição de alguns produtos de consumo em larga escala, como água mineral, refrigerantes, leite, jornais, etc;
- leitura de medidores de consumo de água, energia, gás, etc;
- testes topológicos de sistemas de computadores [Mal89];
- minimização de número de vias, no desenho de circuitos VLSI [Bar90]; e
- inspeção de estruturas metálicas utilizando robôs [Ben03.1].

1.6 Organização do Trabalho

O capítulo II deste trabalho é dedicado ao Problema de Roteamento de Arcos. Inicialmente são definidas as terminologias usadas com maior freqüência. É omitida nessa parte a maioria das definições e termos familiares relacionados à Teoria de Grafos. O Problema do Carteiro Chinês, na sua versão clássica é apresentado e sua solução discutida. Em seguida é feito um estudo taxonômico dos problemas de roteamento de arcos, acompanhado de uma revisão bibliográfica, identificando as soluções mais significativas encontradas na literatura para cada caso.

Como o Problema do Carteiro Chinês no grafo misto é um dos focos de atenção nesse trabalho, o caso é discutido com maior profundidade no capítulo III. Inicialmente o problema é introduzido e as condições de unicursalidade estabelecidas. Em seguida é feita uma revisão bibliográfica para o caso, estudando as abordagens exatas e aproximadas mais relevantes encontradas na literatura.

O mesmo tratamento específico é dado ao Problema do Carteiro Rural, o qual é discutido no capítulo IV, destacando alguns trabalhos recentes sobre o problema.

O capítulo V é dedicado à solução proposta nesse trabalho. Inicialmente o método é apresentado numa forma simplificada para o caso particular do Problema de Carteiro Chinês Misto. É discutida também a solução do PCV, a qual constitui uma das etapas principais do método proposto. A solução proposta é testada com grafos pseudo-manhattan, gerados aleatoriamente. Numa das baterias, os testes são voltados para grafos de porte relativamente grande; na outra, são feitos em grafos pequenos, para possibilitar uma comparação com alguns testes reportados na literatura.

O capítulo VI trata da generalização do método proposto no capítulo anterior para o Problema Geral de Roteamento – PGR. As restrições nos vértices, sua importância e relevância são estudadas e uma metodologia para sua implementação é apresentada. Como exemplo de uma instância do PGR, o capítulo relata testes computacionais para o Problema de Carteiro Rural Misto com Conversões Penalizadas, e compara os resultados com os obtidos por alguns métodos de destaque, registrados na literatura.

O capítulo VII aborda a aplicação do método proposto no problema de coleta de lixo. Algumas restrições adicionais que surgem nesse problema são discutidas, e formas de tratamento apresentadas. É relatada uma experiência com os dados reais de coleta de lixo de um bairro da cidade de Florianópolis.

No capítulo VIII são apresentadas as principais conclusões, frutos da pesquisa realizada, e comentadas as potencialidades do método proposto, seguidas de uma discussão sobre suas limitações.

O capítulo IX relaciona a bibliografia referenciada nesse trabalho.

II – Problemas de Roteamento de Arcos

2.1 Origens

Possivelmente o registro mais antigo de algum problema relacionado com o roteamento de arcos é o famoso enigma das pontes de Konigsberg. Esta cidade, atualmente chamada de Kaliningrad, é construída em ambas as margens do rio Pregel e sobre duas ilhas situadas no rio. As margens do rio e as duas ilhas são conectadas por sete pontes, como mostra a figura 2.1. O problema posto no século XVIII era determinar se existe um roteiro fechado, de modo que alguém pudesse atravessar todas as pontes exatamente uma vez. O problema foi resolvido em 1736 pelo matemático suíço Leonhard Euler que demonstrou a inexistência de tal roteiro [Eul36].

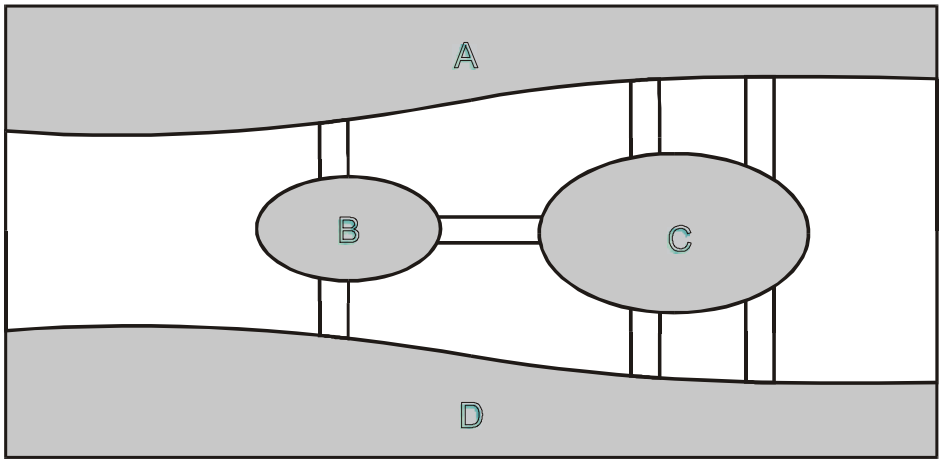


Figura 2.1. As pontes de Konigsberg

De fato, a figura 2.1 pode ser sintetizada pelo grafo mostrado na figura 2.2, no qual os territórios marginais do rio e as ilhas são representados pelos nós, e as pontes pelas arestas. Conforme a argumentação de Euler, num roteiro viável, cada vez que se entra num nó, por meio de qualquer aresta, tem que sair do mesmo, usando uma outra aresta. Logo, o número de arestas incidentes em cada nó deve ser par. Ele concluiu que esta condição, além de ser necessária, é também suficiente.

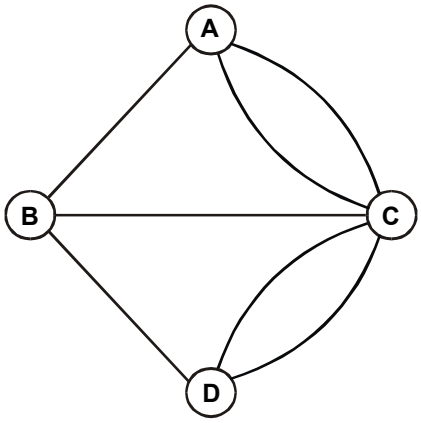


Figura 2.2. Grafo equivalente às pontes de Konigsberg

Devido ao trabalho de Euler, um circuito que contém todos os links de um dado grafo é conhecido como ***Circuito Euleriano***, e o grafo que contém tal circuito, de ***Grafo Euleriano***, ou ***Grafo Unicursal***.

Mais recentemente, em 1962, o matemático chinês Guan Meio-Ko da Escola Normal de Shangtun definiu um problema intimamente relacionado com o problema tratado por Euler [Gua62]. Durante a revolução cultural chinesa, quando ele serviu por algum tempo numa agência dos correios, ficou interessado em achar roteiros de distância mínima para os carteiros. Um carteiro parte de uma agência dos Correios e tem de passar por todas as ruas de uma área de sua responsabilidade, para distribuir as cartas, e voltar ao ponto de partida. A pergunta é: qual o roteiro fechado de distância mínima que cobre todas as ruas a serem servidas, isto é, passa por todos os segmentos de rua, pelo menos uma vez ?

Ao contrário do problema de Euler, que se preocupa com a existência de um circuito euleriano, que passa exatamente uma vez por cada aresta, a questão abordada por Guan era em relação aos grafos não-eulerianos, na tentativa de identificar o circuito de distância mínima que passa pelo menos uma vez por cada aresta, num grafo qualquer. Se o grafo for unicursal, a rede contém um circuito euleriano, que atende perfeitamente à necessidade do carteiro. Caso contrário, a questão se torna um relevante problema combinatorial.

Embora Guan não conseguisse achar uma solução satisfatória para o problema, o seu trabalho provocou interesse geral na comunidade científica do ocidente, e vários outros matemáticos e pesquisadores operacionais começaram a estudar o problema. Devido ao seu pioneirismo, o problema passou a ser denominado como o ***Problema de Carteiro Chinês***.

2.2 Definições

A maioria das definições e termos relacionados à Teoria de Grafos usada nesse trabalho é padrão e pode ser encontrada nos trabalhos clássicos, a exemplo de Christofides [Cri75]. Entretanto, introduzir-se-ão aqueles termos que serão usados com freqüência maior, ou que não são padrões.

Nesse trabalho uma rede é representada por um ***grafo*** $G = (N, A, E)$, em que $N=\{x_1,x_2,..., x_n\}$ representa o conjunto de ***nós*** (ou ***vértices***), $A=\{a_1,a_2,...,a_r\}$ o conjunto de ***arcos***, e $E=\{e_1,e_2,...,e_m\}$ o conjunto de ***arestas***. $n = |N|$, $r = |A|$ e $m = |E|$ representam a cardinalidade de cada conjunto. Refere-se ao conjunto maior, definido por $L = A \cup E$, como o conjunto de ***links*** do grafo. Portanto, um link $l \in L$ pode ser um arco, ou uma aresta.

Um link pode ser descrito pelo par de nós (x_i, x_j) que indicam seus nós ***terminais***. Em se tratando de uma aresta, a ordem de nós terminais, nesta notação, é irrelevante; porém, no caso de arcos a ordem é do nó ***inicial*** para o ***final***. Sempre que for conveniente, se pode considerar uma aresta (x_i, x_j) como um par de arcos contrariamente orientados (x_i, x_j) e (x_j, x_i). Dois nós conectados por um link são chamados de ***adjacentes***. A cada link (x_i, x_j) de um grafo pode ser associado um ***custo*** d_{ij}. Uma matriz $D = [d_{ij}]$ é a *matriz de custos* associada ao grafo, onde d_{ij} é o custo do link $(x_i, x_j) \in L$, e $d_{ij} = \infty$ se $(x_i, x_j) \notin L$.

Se $E=\varnothing$, então G é um ***grafo orientado***; se $A=\varnothing$, ele é um ***grafo não-orientado***; e se $E\neq\varnothing$ e $A\neq\varnothing$, o grafo é denominado de ***grafo misto***.

Quando uma malha urbana ou uma rede rodoviária é representada por um grafo, os arcos representam os trechos de ruas de mão única, e as arestas os de mão dupla. Os nós são os cruzamentos que permitem passagem de um trecho de rua para outro. Todavia, um grafo é uma estrutura mais genérica do que uma malha urbana.

Uma das formas particulares de um grafo é o ***grafo manhattan***, o qual tem a forma de uma grade reticulada, que representa uma malha urbana regular (quadras retangulares). Uma outra é o ***grafo pseudo-manhattan***, que tem a forma do grafo anterior, com a liberdade de existência de links dispostos diagonalmente entre os vértices dos retângulos.

Um ***grafo completo*** é um grafo orientado (ou não-orientado) em que para quaisquer nós $x_i \in N$ e $x_j \in N$ o arco $(x_i, x_j) \in A$ (ou a aresta $(x_i, x_j) \in E$).

Num grafo não-orientado, para cada nó x_i define-se o ***grau*** $g(x_i)$ como o número de arestas que incidem no nó x_i. Quando o grafo é orientado, define-se como o ***grau de entrada***, $g_e(x_i)$, o número de arcos cujos nós finais são o nó x_i. Analogamente, o ***grau de saída*** $g_s(x_i)$, é o número de arcos, cujos nós iniciais são o nó x_i. É fácil verificar que a soma de graus de entrada de todos os nós de um grafo é igual a soma dos graus de saída.

Um ***caminho*** é uma seqüência de links, respeitando sua orientação, onde o nó final de um é o inicial do próximo. Desta forma um caminho $C(x_i, x_j)$ começa em um nó x_i e termina em x_j, onde x_i e x_j não são necessariamente adjacentes. x_i e x_j são considerados os nós inicial e final do caminho, respectivamente, e conectados nesta ordem pelo caminho. Uma ***cadeia*** é uma seqüência de links, também ligando dois nós não necessariamente adjacentes, sem respeito à orientação dos arcos.

Um ***circuito*** é um caminho em que o nó inicial coincide com o nó final.

Um grafo é dito ***fortemente conexo***, se para qualquer par ordenado de nós x_i, x_j existe pelo menos um caminho que conecta x_i a x_j. Essa definição implica que num grafo fortemente conexo, dois nós quaisquer são mutuamente acessíveis. Um grafo é dito ***conexo***, ou ***fracamente conexo***, se para qualquer par de nós x_i, x_j existe pelo menos uma cadeia que conecta x_i a x_j. Se pelo menos para um par de nós tal cadeia não existe, o grafo então é ***desconexo***.

Dado um grafo $G=(N, L)$, um ***grafo parcial*** G_p de G é o grafo (N, L_p) com $L_p \subset L$. Portanto, um grafo parcial é um grafo com o mesmo número de nós, porém com apenas um subconjunto próprio de links do grafo original.

Um ***subgrafo*** G_s de G é o grafo (N_s, L_s) com $N_s \subset N$, e $L_s = \{(x_i, x_j) \mid (x_i, x_j) \in L\ ,\ x_i \in N_s,\ x_j \in N_s\}$. Portanto, um subgrafo é um grafo contendo um subconjunto de nós, porém com todos os links que conectam estes nós no grafo original.

Num grafo $G=(N, L)$, um ***componente*** é definido como um subgrafo $K=(N_K, L_K)$, em que K é conexo e não existe nenhum link (x_i, x_j) tal que $x_i \in N_K$ e $x_j \in N \setminus N_K$. Se G é

conexo, então ele é formado por um único componente; um grafo desconexo é formado por mais de um componente.

Um circuito que passa por todos os nós de um grafo, sem que repita o mesmo nó mais de uma vez, é denominado de ***Circuito Hamiltoniano***. Nem todo grafo contém um circuito hamiltoniano; porém quando possui, é chamado de ***Grafo Hamiltoniano***.

Um circuito que passa por todos os links de um grafo, sem que repita o mesmo link mais de uma vez, é denominado de ***Circuito Euleriano***. Também, nem todo grafo contém um circuito euleriano, e quando possui, ele é chamado de ***Grafo Euleriano***, ou ***Grafo Unicursal***. Um circuito que passa por todos os links de um grafo, pelo menos uma vez, é denominado de ***Circuito de Carteiro***. Todo grafo fortemente conexo contém um circuito de carteiro.

Uma ***α-aproximação*** de um problema é uma solução cujo custo, no pior caso, é $(1+\alpha)$ vezes a solução ótima.

2.3 Circuitos Eulerianos

Como foi definido acima, dado um grafo não-orientado $G=(N, E)$, fortemente conexo, um circuito que contém todas as arestas do grafo, sem que repita a mesma aresta mais de uma vez, é denominado de circuito euleriano. E como foi visto, nem todo grafo contém um circuito euleriano; quando possui, ele é chamado de grafo euleriano, ou grafo unicursal. O teorema básico sobre a existência de um circuito euleriano, em um grafo não-orientado, é o seguinte:

TEOREMA [Eul36] - Um grafo fortemente conexo $G=(N, E)$ contém um circuito euleriano, se, e somente se, o grafo não tem nenhum nó de grau ímpar.

A necessidade da condição estabelecida nesse teorema pode ser verificada, tendo em vista que qualquer circuito euleriano deve usar uma aresta para entrar em cada nó, e uma outra para partir do mesmo. Considerando que todas as arestas devem ser percorridas uma única vez, então deve haver um número par de arestas incidentes a cada nó. Isso é, o grau de todo nó deve ser par.

A suficiência da condição poderá ser estabelecida, usando uma prova construtiva. É fácil verificar que se G é fortemente conexo e todos os seus nós são de grau par, nele se pode construir um circuito euleriano.

Uma forma de construir um circuito euleriano é começar o traçado em qualquer nó arbitrário x_i, percorrer uma aresta não utilizada para chegar a um outro nó x_j, e a partir deste prosseguir para um outro nó, repetindo o processo. Como o grau de todos os nós é par, então cada vez que se entra num nó, deve haver uma aresta não utilizada, disponível para sair. Portanto, o processo terminaria forçosamente no mesmo nó de partida x_i, quando se completa um circuito C. Se C contiver todas as arestas, então um circuito euleriano é obtido; caso contrário, identifica-se o grafo parcial de G_p, formado pelas arestas não utilizadas do grafo G. Pode-se verificar que todos os nós de G_p também seriam de grau par. Escolhe-se um nó x_k que é terminal de uma aresta de G_p, tal que x_k esteja contido em C. Tal nó existe, pois se não, G não seria conexo. A partir de x_k será construído um novo circuito C', o qual será aninhado a C, a partir de x_k, formando um único circuito. Este procedimento será continuado, até a obtenção de um circuito euleriano completo no grafo G.

Edmonds [Edm73] utilizou uma abordagem diferente, baseada em construção de uma arborescência, para construir um circuito euleriano. Seja qual for o método, quando o grafo é euleriano, a obtenção de um circuito euleriano é uma tarefa trivial.

Uma propriedade notável de um grafo, quando não é euleriano, é a de possuir um número par de nós de grau ímpar. Este fato pode ser explicado pelo número total de incidências nos nós do grafo: se o grafo possui m arestas e cada aresta está em contato com dois nós, então a soma de graus de todos os nós é $2m$, que é um número par. Seja g_p a soma de graus dos nós de grau par, e g_i a soma para os nós de grau ímpar. Portanto:

$$g_p + g_i = 2m$$

Obviamente g_p é um número par, logo g_i deve ser um número par. Se a soma de alguns números ímpares é par, é porque a quantidade dos números ímpares é par.

Logo, pode-se afirmar que em qualquer grafo há um número par de nós de grau ímpar.

2.4 Problema de Carteiro Chinês

Dado um grafo não-orientado $G=(N, E)$, conexo, onde a cada aresta $e \in E$ é associado um custo d_e, o ***Problema de Carteiro Chinês – PCC***, consiste em achar um circuito de custo mínimo, contendo todas as arestas, pelo menos uma vez (circuito de carteiro).

Se o circuito de carteiro contiver cada aresta exatamente uma única vez, logicamente o circuito seria de custo mínimo. Como foi visto, tal circuito apenas existe num grafo euleriano. Se alguns nós são de grau ímpar, então qualquer circuito que percorre todas as arestas do grafo pelo menos uma vez, percorrerá algumas delas mais de uma vez. O problema, nesse caso é minimizar o custo total das passagens adicionais.

A solução consiste em inserir apropriadamente cópias de algumas arestas, de modo que o ***grafo aumentado*** (grafo original, acrescido pelas arestas copiadas) seja euleriano e o custo total das cópias mínimo.

Guan [Gua62] demonstrou que numa solução ótima de carteiro nenhuma aresta precisa ser percorrida mais de duas vezes. De fato, se o grafo aumentado contém mais de uma cópia de algumas arestas, as cópias em excesso associadas a cada aresta podem ser eliminadas em número par, sem que isso modifique a paridade de grau dos nós afetados, nem afetar a unicursalidade do grafo.

A estratégia para resolver o PCC num grafo não orientado consiste numa tentativa de tornar par o grau dos nós que estão na situação ímpar, mantendo inalterado o grau dos demais nós. A figura 2.3 mostra um grafo originalmente não-euleriano, por conter dois nós de grau ímpar. A duplicação das arestas num caminho que liga estes dois nós torna o grafo euleriano. Numa solução ótima, entretanto, tais duplicações devem acontecer ao longo do caminho mais curto, o que não acontece na solução apresentada na figura 2.3.

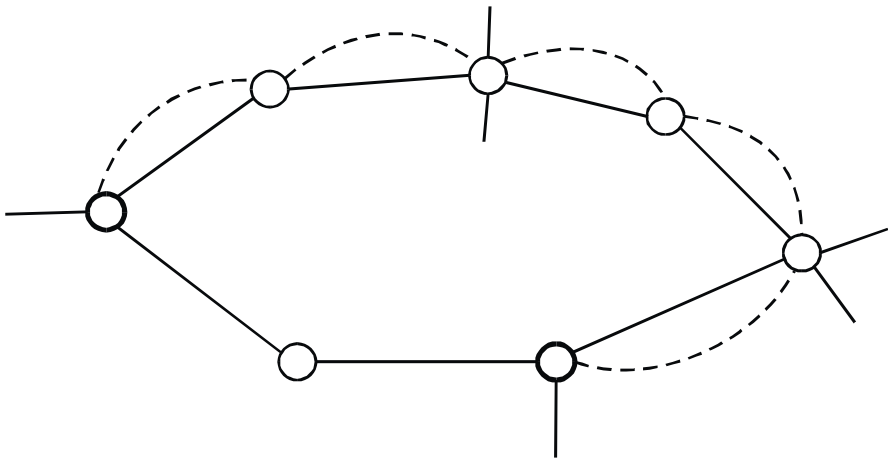


Figura 2.3.

Quando o grafo tem mais que dois nós de grau ímpar, a duplicação de caminhos mínimos que ligam pares de nós de grau ímpar, embora necessário, não é suficiente para garantir a otimalidade da solução. Figura 2.4 (a) mostra uma situação em que dois pares de nós de grau ímpar estão conectados por meio de dois caminhos mínimos duplicados. Entretanto, os pares de nós não estão associados corretamente. Uma solução melhor é aquela mostrada na figura 2.4 (b).

Este último ponto é o aspecto combinatorial mais relevante do problema, e pode ser formulado como o ***emparelhamento perfeito***, cuja solução foi provida por Edmonds [Edm73]. Portanto, o algoritmo para o PCC num grafo não-orientado pode ser resumido como o seguinte:

ALGORITMO PARA O PCC

0) Dado um grafo não-orientado $G = (N, E)$, fortemente conexo, onde a cada aresta $e \in E$ é associado um custo d_e;
1) Identifique o conjunto $I \subseteq N$ de nós de grau ímpar. Se $I = \varnothing$, então G é euleriano. Faça $G_e = G$, e vá ao passo 5;
2) Construa o grafo completo $G_I = (I, C)$, onde C é o conjunto de caminhos mínimos entre todos os pares de nós de I, calculados no grafo G;
3) Encontre o emparelhamento de custo mínimo em G_I;
4) Identifique os caminhos mínimos $C^* \subseteq C$ em G, os quais ligam os nós do emparelhamento de mínimo custo. Seja $E^* \subseteq E$ o conjunto de todas as arestas que formam os caminhos mínimos em C^*. O grafo aumentado $G_e = (N, E \cup E^*)$ é euleriano;

5) Construa um circuito euleriano em G_e.

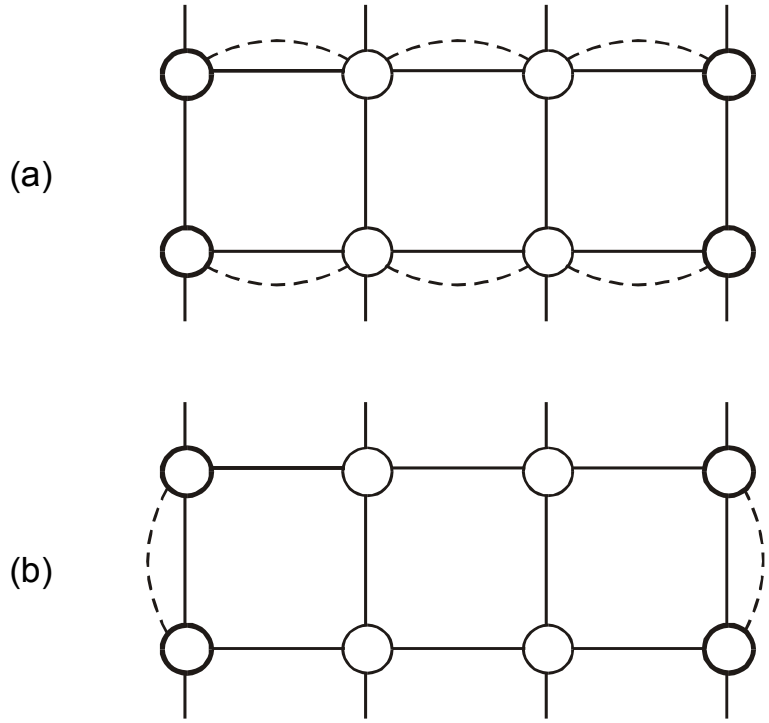


Figura 2.4

A rotina de emparelhamento perfeito do passo 3, no pior caso requer computações na ordem de n^3 operações aritméticas. Igualmente, os cálculos de caminhos mínimos no passo 2 envolvem operações na mesma ordem. Portanto, o algoritmo acima tem uma complexidade global da ordem n^3 [Law76].

2.5 Taxonomia dos Problemas de Roteamento de Arcos

Como foram examinados na Introdução, os problemas de roteamento de veículos são sensíveis a vários grupos de restrições relacionadas com instalações, veículos, demanda e a rede. Cada caso acaba tendo características particulares que inviabilizam a adoção de uma solução genérica para o problema. Entretanto, os métodos particulares, na sua composição, em geral usam as rotinas que resolvem os problemas básicos de roteamento. O sucesso na solução do problema particular, depende em grande parte da qualidade de soluções existentes para os problemas simplificados. Estes problemas fundamentais são aqueles

formulados e tentados pelos pesquisadores operacionais ao longo dos últimos anos, e que na sua grande maioria continua sendo desafios para a nova geração.

Mesmo simplificando os problemas de roteamento e desconsiderando grande parte das restrições com relação a instalações e veículos, ignorando o parâmetro da capacidade no atendimento à demanda, há uma variedade de formulações célebres, abordadas na literatura. O objetivo aqui é relacionar e comentar as mais relevantes.

Seja $G = (N, A, E)$ um grafo misto, fortemente conexo. Suponha que a demanda esteja localizada nos subconjuntos $N' \subseteq N$, $A' \subseteq A$ e $E' \subseteq E$. Isto significa que no circuito a ser construído todos estes subconjuntos devem ser visitados, mas não necessariamente todos os elementos dos conjuntos *N*, *A*, e *E*. Denominam-se os subconjuntos *N'*, *A'*, e *E'* como os *conjuntos requeridos*. É considerado ainda que existe apenas um veículo, sem limite de capacidade, para atender a toda a demanda, e uma única instalação, sem limite de atendimento à demanda, a qual está localizada num dos nós de *N'*. Alguns dos problemas mais tratados na literatura são os seguintes:

2.5.1- Problema de Caixeiro Viajante -PCV

Se $N' = N$ e $A' = E' = \varnothing$, o problema de achar um circuito de custo mínimo em *G* é reduzido ao clássico *Problema de Caixeiro Viajante – PCV*, o qual é um dos problemas mais estudados em otimização combinatória. O problema é estudado em duas versões principais: caso simétrico, em que $A = \varnothing$, e o assimétrico, quando $A \neq \varnothing$. Karp [Kar72]tem demonstrado que PCV é NP-completo em todas as suas versões. Isso implica na inexistência de algoritmos polinomialmente limitados para solução exata do problema. Nas abordagens exatas, as tentativas têm esbarrado no excesso de tempo de execução, ou no excesso de armazenagem de informações. Um trabalho de destaque nessa linha é de Applegate et al. [App95].

Devido às dificuldades com relação à eficiência computacional dos métodos exatos, muitas soluções heurísticas têm sido desenvolvidas. Johnson et al. [Joh97] e Voudouris et al. [Vou99] têm sugerido soluções aproximadas que foram aplicadas com sucesso aos PCV's de grande porte. A meta-heurística denominada de *Busca Local Dirigida*, desenvolvida pelos últimos autores, tem encontrado soluções próximas a ótima para problemas com mais de 10.000 nós, em tempos relativamente curtos [Rod00].

A menção do PCV aqui foi devido a sua importância no contexto de problemas de roteamento. Todavia, o PCV não é um Problema de Roteamento de Arcos.

2.5.2- Problema de Carteiro Chinês

Quando $N' = \varnothing$, $A = \varnothing$ e $E' = E$, a questão se torna o Problema de Carteiro Chinês, na sua forma clássica, proposta por Guan [Gua62]. O problema é resolvido por meio de um algoritmo polinomial de complexidade n^3, desenvolvido por Edmonds [Edm73], o qual foi discutido anteriormente.

Um problema intimamente ligado ao PCC, é o de cobertura de um grafo por meio de circuitos não-orientados (ciclos). O objetivo é cobrir todas as arestas do grafo por ciclos simples, de tal maneira que o comprimento total da cobertura seja mínimo. Este problema surge nos projetos de sistemas de irrigação e na análise de circuitos elétricos. Itai et al. [Ita81] provam que minimizando o comprimento da cobertura e resolvendo PCC não são equivalentes. Kesel'man [Kes87] provou que nos grafos planares, os dois problemas são equivalentes.

2.5.3- Problema de Carteiro Chinês Orientado - PCCO

Se $N' = \varnothing$, $A' = A$ e $E = \varnothing$, tem-se o caso orientado do Problema de Carteiro Chinês. Nesse caso, um circuito de carteiro de custo mínimo precisa ser construído num grafo completamente orientado.

A condição necessária e suficiente para existência de um circuito euleriano num grafo orientado é que, além de ser fortemente conexo, o grafo deve ser *simétrico*. Isso é, para cada nó o grau de entrada deve ser igual ao grau de saída. Quando em alguns nós o número de arcos de entrada diverge do número de arcos de saída, o grafo não é unicursal e, para torná-lo assim é necessário acréscimo de cópias apropriadas de alguns arcos.

Beltrami e Bodin [Bel74] mostram que um grafo euleriano de custo mínimo pode ser construído resolvendo um simples Problema de Transporte. Nesse caso, os nós com excesso de entrada serão considerados como *suprimento* e os com excesso de saída como *demanda*. A solução do Problema de Transporte indica qual nó de suprimento deva ser associado a qual de demanda. As cópias dos arcos devem ser acrescentadas ao grafo, ao

longo dos caminhos mínimos que ligam os nós de suprimento aos de demanda na solução do Problema de Transporte.

A complexidade computacional desse algoritmo é da ordem $O(mn^2)$. Edmonds e Johnson [Edm73] apresentam um algoritmo alternativo que corresponde à solução de fluxo de custo mínimo no grafo, cuja complexidade é de $O(n^3)$. Lin e Zhao [Lin88] apresentam uma abordagem diferente, desenvolvida na base do Teorema de Folga Complementar da Programação Linear. A complexidade desta é de $O(kn^2)$, onde k depende da estrutura do grafo. Eles demonstram que para os grafos esparsos k é menor que m e n, portanto para estes o método funciona melhor que os outros mencionados acima.

Um problema interessante associado a PCCO consiste em determinar um *subgrafo euleriano de custo máximo* num dado grafo orientado. Richey e Parker [Ric91] têm demonstrado que este é um problema NP-hard no caso genérico, e sugerem um algoritmo que resolve o problema num caso particular.

2.5.4- Problema de Carteiro Chinês Misto - PCCM

Esta formulação é obtida a partir de $N' = \varnothing$, $A' = A$ e $E' = E$. O PCCM é a versão do PCC que mais se aproxima à realidade das malhas urbanas, não obstante, é a mais difícil do ponto de vista da solução.

Papadimitriou [Pap76] demonstrou que PCCM é NP-completo. Portanto, não existem soluções exatas com complexidade polinomial para este caso. O caminho natural é via soluções aproximadas. Entretanto, poucos métodos aproximados foram desenvolvidos para o caso.

Pelo fato de que um dos focos de atenção desta tese é o PCCM, o Capítulo III será dedicado a uma revisão bibliográfica particular e análise das soluções existentes para o caso.

2.5.5- Problema de Carteiro com Vento - PCCV

Existe uma formulação semelhante a PCC, com $N' = \varnothing$, $A = \varnothing$ e $E' = E$, mas com um grau de liberdade a mais: cada aresta pode ser percorrida em ambos os sentidos com custos desiguais. Esta situação pode ocorrer quando, por exemplo, um dos sentidos é a

subida de uma ladeira, e o outro a descida no sentido contrário; ou, um dos sentidos é a favor do vento, e o outro contra.

O PCCV foi introduzido inicialmente por Minieka [Min79] e recentemente recebeu atenção de alguns pesquisadores. É considerada uma versão interessante do PCC, não pela sua real aplicação, mas porque todos os três casos anteriores de PCC podem ser derivados desta simples generalização. Por exemplo, um PCCM pode ser formulado como PCCV; basta fixar para as arestas custos iguais, para ambos os sentidos, e para os arcos considerar o custo normal no sentido da orientação do arco, e custo igual a ∞ no sentido contrário.

Guan [Gua84] provou que computacionalmente PCCV é equivalente a PCCM, portanto é NP-completo. Ao mesmo tempo, ele sugeriu uma solução aproximada para o caso: em cada aresta o custo é convertido para o valor médio das duas orientações. O problema resultante é resolvido como PCC, usando algoritmo de emparelhamento de custo mínimo. O grafo euleriano obtido é decomposto arbitrariamente em sub-ciclos não orientados. Em seguida, cada sub-ciclo é orientado escolhendo a orientação que produz o menor custo. Pearn e Li [Pea94] mostraram que esta é uma ∞-aproximação para o PCCV. Isto é, no pior caso, a razão (Solução de Guan) / (Solução Ótima) tende a ∞.

Win [Win89] sugeriu uma abordagem semelhante, mas superior à de Guan em que os sub-ciclos não são gerados arbitrariamente, mas sim de forma ótima, resolvendo um problema de fluxo no grafo euleriano inteiro. Esse algoritmo é de 1-aproximação para o PCCV.

Pearn e Li [Pea94] desenvolveram uma versão melhorada para o algoritmo de Guan, a qual reduziu em média 4,2% o custo, em relação ao algoritmo de Guan original. Os autores sugeriram, também, uma nova estratégia para o algoritmo de Win, a qual denominaram de algoritmo de *Win Reverso*. A nova estratégia demonstrou ser mais eficiente que a versão original, apenas quando o grafo é *próximo a ímpar* (predominantemente composto de nós de grau ímpar) e com "forte ventania" (diferença acentuada entre custos das duas orientações das arestas). Os testes computacionais foram realizados em grafos de até 50 nós e 377 arestas.

Grötschel e Win [Gro92] estudaram a estrutura poliedral de PCCV e desenvolveram um algoritmo de planos de corte para o problema, baseado numa descrição linear parcial do poliedro. A solução foi testada em grafos de até 264 nós e 489 arestas. Dos 36 casos

testados, em 31 o método encontra uma solução ótima e inteira; quando o algoritmo falha em achar valores inteiros, eles recomendam uma estratégia de arredondamento para os valores finais.

2.5.6- Problema de Carteiro Rural - PCR

O *Problema de Carteiro Rural* constitui uma classe de problemas de roteamento de arcos em que $N' = \varnothing$, $A' \subset A$ e / ou $E' \subset E$. Ao contrário de todas as versões de PCC, em PCR apenas um subconjunto de links necessita de serviço. Portanto é uma formulação mais genérica e mais realista para os problemas de distribuição que contém o PCC como caso particular. A exemplo do PCC, o PCR pode ser formulado num grafo não-orientado, num grafo orientado – PCRO, ou num grafo misto – PCRM.

Ao contrário do PCC, o PCR é NP-hard, mesmo nos casos orientado e não-orientado. Pela importância do PCR para os problemas reais de roteamento, e por fazer parte dos objetivos desta tese contribuir à sua solução, o capítulo IV será dedicado ao seu estudo.

2.5.7- Problema da Empilhadeira – PE

O problema da empilhadeira (Stacker Crane) é um caso particular de PCR, definido num grafo $G = (N, A, E)$, com $A' = A$, $E \neq \varnothing$, $E' = \varnothing$ e $N' = \varnothing$. O problema é determinar, num grafo misto, o menor circuito que inclui todos os arcos, enquanto nenhuma aresta é requerida. Os arcos podem ser vistos como os movimentos a serem executados por uma empilhadeira, cada um exatamente uma vez, numa direção específica. Se os custos dos arcos forem nulos, o problema se reduz a um PCV, portanto, PE é um problema NP-hard.

Frederickson et al. [Fre78] propuseram duas heurísticas em que é preciso que o grafo *G* satisfaça duas condições:

i) cada nó é incidente por pelo menos um arco de *A*, e

ii) os custos aplicados às arestas satisfazem a desigualdade triangular.

Se *G* não satisfaz estas duas propriedades, ele pode ser transformado num grafo equivalente que satisfaça as mesmas. As duas heurísticas são denominadas de *Largearcs* e *Smallarcs*. A primeira apresenta resultados melhores, quando o custo total dos arcos é

grande, comparado com o custo total numa solução ótima. A segunda funciona melhor numa situação inversa.

A complexidade de ambas as heurísticas é $O(\max\{|N|^3, |A|^3\})$. O método Largearcs produz uma rota r para o PE, cujo comprimento $d_L(r)$ satisfaz

$$d_L(r) \leq 3z^* - 2d(A),$$

onde z^* é o valor da solução ótima de PE e $d(A)$ é a soma de custos de todos os arcos de A. Para o método Smallarcs, o comprimento $d_S(r)$ da rota gerada satisfaz

$$d_S(r) \leq \frac{1}{2}(3z^* + d(A)).$$

Uma forma para melhorar o pior caso desses algoritmos é a aplicação de ambos e escolhendo o melhor resultado. Desta forma, o pior caso se reduz para

$$d(r) \leq 9z^*/5.$$

2.5.8- Problema Geral de Roteamento - PGR

Uma formulação possível para os problemas de roteamento é quando $N' \neq \varnothing$, $A' \subset A$ e $E' \subset E$, conhecida como o problema geral de roteamento. A diferença entre PGR e PCRM é o fato de que no PGR, além de nós implicitamente requeridos, isso é, os nós que incidem aos links requeridos, eventualmente outros nós precisam ser servidos. É o caso mais genérico entre todas as formulações não-capacitadas de problemas de roteamento. Portanto, todos os casos acima estudados podem ser formulados como casos especiais do PGR. O PGR foi proposto e estudado inicialmente por Orloff [Orl74].

Embora se possa formular problemas reais de distribuição na forma do PGR, a sua importância não é devida a uma possível aplicação direta; se houver algum método eficiente para este caso, isso significaria a existência de uma ferramenta comum para a solução de uma variedade de formulações particulares do problema. Entretanto, não houve progresso significativo na solução desse problema.

Vale notar também que o PGR não é precisamente um Problema de Roteamento de Arcos, embora estes possam ser formulados como tal.

III – Problema do Carteiro Chinês Misto

3.1 Circuitos Eulerianos em Grafos Mistos

CONDIÇÕES DE UNICURSALIDADE

Nessa seção trabalha-se com um grafo misto fortemente conexo $G = (N, A, E)$. A cada link $l \in A \cup E$ é associado um custo não-negativo d_l. Os conjuntos requeridos são $N' = \varnothing$, $A' = A$ e $E' = E$. Por conveniência são usadas as notações A e E, também para assinalar os conjuntos requeridos.

O PCCM consiste em achar um circuito de custo mínimo em *G*, contendo todos os arcos em *A* e todas as arestas em *E*. Como foi visto, quando o grafo é euleriano, é garantida a existência de um circuito que contém todos os links, exatamente uma vez. Tal circuito, denominado de circuito euleriano, obviamente seria de custo mínimo. As condições necessárias e suficientes para a unicursalidade de um grafo misto foram estabelecidas por Ford e Fulkerson [For62], as quais são:

a) *G* seja fortemente conexo;
b) Em cada nó deve incidir um número par de links;
c) Para qualquer $X \subset N$, a diferença entre o número de arcos orientados de *X* a $N - X$ e o número de arcos orientados de $N - X$ a *X* deva ser menor, ou igual, ao número de arestas que conectam *X* e $N - X$.

A última condição freqüentemente é chamada de *Condição de Conjunto Balanceado*. O grafo que satisfaz a condição (b) é chamado de *grafo par*; quando para cada nó o número de arcos de entrada é igual ao número de arcos de saída, ele é chamado de *simétrico*, e quando satisfaz as condições de conjuntos balanceados, é denominado de *grafo balanceado*.

Pelas definições acima, se o grafo *G* é par e balanceado, então é euleriano. Se um grafo é par e simétrico, também é euleriano, pois essas duas condições juntas implicam no balanceamento do grafo. Entretanto, a simetria não é uma condição necessária para a unicursalidade.

Se o grafo misto G é euleriano, o traçado de um circuito euleriano compreenderá as seguintes etapas:

a) assinalar orientações para algumas arestas de G, de modo a torná-lo simétrico;
b) orientar as demais arestas de G em forma de circuitos; e
c) determinar um circuito euleriano no grafo orientado resultante.

Ford e Fulkerson [For62] apresentam um procedimento para a etapa (a), para transformar um grafo não-simétrico em simétrico, o qual é descrito abaixo:

ALGORITMO PARA TRANSFORMAR UM GRAFO MISTO EM SIMÉTRICO

P1) Substitua cada aresta de G por um par de arcos contrariamente orientados, obtendo o grafo completamente orientado $G' = (N, A')$. Assinale a cada arco de $A' \cap A$ um limite inferior de fluxo igual a 1, e a cada arco de $A' - A$ o limite inferior igual a 0. Também, atribuir a cada arco de A' um limite superior de fluxo igual a 1;

P2) Usando um algoritmo de fluxo, determine uma circulação viável em G'. Seja f_{ij} o fluxo no arco (x_i, x_j);

P3) Orientar algumas arestas de G como segue: se $(x_i, x_j) \in E$, $f_{ij} = 1$ e $f_{ji} = 0$, oriente (x_i, x_j) de x_i para x_j.

A aplicação deste algoritmo a um grafo misto qualquer resulta em um grafo simétrico, porém poderão persistir arestas não orientadas. Caso o grafo seja euleriano, tais arestas aparecerão em ciclos, os quais poderão ser orientados usando o algoritmo a seguir [Eis95.1], o qual é um detalhamento da etapa (b) do procedimento de construção de um circuito euleriano:

ALGORITMO PARA ORIENTAÇÃO COMPLETA DE UM GRAFO SIMÉTRICO

P1) Se todas as arestas estão orientadas, pare.

P2) Seja x_i um nó terminal de uma aresta não-orientada (x_i, x_j). Fixe $u = x_i$, e $v = x_j$.

P3) Oriente a aresta (u, v) de *u* para *v*. Se $v = x_i$, volte ao passo 1.

P4) Fixe $u = v$ e identifique uma nova aresta não-orientada (u, v), tendo *u* como um de seus terminais. Volte ao passo 3.

Uma vez obtido um grafo completamente orientado, na etapa seguinte (c) um circuito euleriano é construído. Aardenne-Ehrenfest e Bruijn sugerem o seguinte procedimento, o qual está contido em [Eis95.1]:

ALGORITMO PARA CONSTRUÇÃO DE UM CIRCUITO EULERIANO NUM GRAFO ORIENTADO

P1) Construa uma árvore geradora enraizada em um nó qualquer n_r.

P2) Rotule todos os arcos, como segue: ordene e rotule os arcos que saem de n_r num modo arbitrário; em seguida ordene e rotule os arcos que saem de outros nós sucessivamente, num modo arbitrário, contanto que o último arco seja o usado na árvore.

P3) Escolhendo um nó arbitrário *n*, inicie a construção de um circuito euleriano, trilhando pelo arco de menor rótulo que emana do nó *n*; sempre que se entra num novo nó, o mesmo é deixado através do arco não usado de menor rótulo. O procedimento termina, quando todos os arcos são usados.

Como foi observado por Ford e Fulkerson [For62], seu algoritmo de fluxo pode ser aplicado a um grafo par, com unicursalidade a priori não conhecida. O algoritmo irá produzir um fluxo viável, que afirma a unicursalidade; ou irá falhar, o que significa que o grafo não é unicursal.

Se *G* não é unicursal, um *grafo ampliado* G_a que satisfaça as condições de unicursalidade pode ser definido. Isso significa acréscimo apropriado de cópias de alguns arcos e de algumas arestas a *G* de modo que o grafo resultante seja euleriano. Neste caso, resolver o PCCM é equivalente a achar o acréscimo de custo mínimo que torna o grafo euleriano.

O PCCM é a versão mais difícil do problema de carteiro chinês, do ponto de vista de soluções. Papadimitriou [Pap76] provou que ele é NP-completo. A seguir, examinam-se as tentativas de solução encontradas na literatura.

3.2 Soluções Exatas

3.2.1 Uma Formulação Básica

As soluções exatas para o PCCM, em geral, são baseadas em formulações de programação linear inteira. Kappauf e Koehler [Kap79] apresentam uma formulação básica para o problema, a qual é semelhante à descrita abaixo:

Seja $\delta(i)$ o conjunto de links (v_i, v_j) incidentes no nó v_i, e $L(S)$ como o conjunto de links (v_i, v_j) com $v_i \in S$, $v_j \notin S$, onde $S \subset N$. Seja também x_{ij} uma variável inteira indicando quantas vezes o link (v_i, v_j) é percorrido de v_i a v_j numa solução ótima de PCCM. O problema então pode ser formulado como:

Minimizar $\sum_{(v_i,v_j)\in E} c_{ij}(x_{ij} + x_{ji}) + \sum_{(v_i,v_j)\in A} c_{ij}x_{ij}$ (1)

sujeito a

$$x_{ij} + x_{ji} \geq 1 \qquad \left((v_i, v_j) \in E\right) \qquad (2)$$

$$x_{ij} \geq 1 \qquad \left((v_i, v_j) \in A\right) \qquad (3)$$

$$\sum_{(v_i,v_j)\in\delta(i)} (x_{ij} - x_{ji}) = 0 \qquad \left(v_i \in N\right) \qquad (4)$$

$$x_{ij} \geq 0 \qquad \left((v_i, v_j) \in A\right) \qquad (5)$$

$$x_{ij}, x_{ji} \geq 0 \qquad \left((v_i, v_j) \in E\right) \qquad (6)$$

$$x_{ij}, x_{ji} \text{ inteiros} \qquad \left((v_i, v_j) \in A \cup E\right) \qquad (7)$$

A restrição (5), embora redundante, é colocada para explicitar uma condição básica do modelo linear. Estes autores não apresentaram nenhum resultado computacional, com base nessa formulação. Entretanto, eles estabelecem uma correspondência um-a-um entre os pontos extremos do poliedro definido por conjunto das restrições (2) a (7) e os grafos

eulerianos associados ao problema. Ralph [Ral93] mostrou que os pontos extremos do poliedro de relaxação linear, definido por (2) a (6) são todos *meio-inteiros*.

3.2.2 Solução de Grötschel e Win

Um exemplo de utilização dessa formulação foi dado por Grötschel e Win [Gro92]. Originalmente, eles desenvolveram um método de solução para o PCCV, baseado numa descrição parcial do seu poliedro de soluções. O método pode ser adaptado para o PCCM da seguinte forma:

Seja P o poliedro de vetores $x = (x_{ij}, x_{ji})$ que satisfazem (2), (3), (4), (5) e (6). Então G é euleriano se, e somente se, cada vértice de P é inteiro. Eles demonstram que os componentes dos vértices de P são sempre 0, $^1/_2$, ou um inteiro positivo. Minieka [Min79] mostrou que em qualquer solução ótima de PCCM, apenas um dos três seguintes casos ocorre para qualquer aresta (v_i, v_j):

i) $x_{ij} = 0$ e $x_{ji} \geq 1$;

ii) $x_{ji} = 0$ e $x_{ij} \geq 1$;

iii) $x_{ij} = x_{ji} = 1$.

Isso sugere um esquema tri-vias de *branching* numa busca enumerativa. Grötschel e Win [Gro92] mostram que para qualquer $|L(S)|$ ímpar, as seguintes *desigualdades de corte ímpar* são válidas:

$$\sum_{(v_i,v_j)\in L(S)} (x_{ij} + x_{ji}) \geq |L(S)| + 1 \qquad S \subset N \qquad (8)$$

$$\sum_{v_i\in S, v_j\notin S} x_{ij} \geq {}^1/_2(|L(S)| + 1) \qquad S \subset N \qquad (9)$$

$$\sum_{v_i\in S, v_j\notin S} x_{ji} \geq {}^1/_2(|L(S)| + 1) \qquad S \subset N \qquad (10)$$

Grötschel e Win resolveram (2), (3), (4), (5) e (6), acrescidas por (8),(9) e (10) por meio de uma rotina de Programação Linear. O papel de desigualdades de corte ímpar é fortalecer a relaxação da restrição (7). Com isso, eles resolveram 8 PCCM's em otimalidade, dentre 8 tentados, sem nenhum branching. Os problemas ficavam na faixa de $52 \leq |N| \leq 172$, $37 \leq |E| \leq 154$, e $31 \leq |A| \leq 116$. O tempo de processamento ficou entre

9 a 84 segundos num NORSK DATA ND-540. O maior problema tentado foi do tipo PCCV com 264 nós e 489 arestas. Das cinco versões deste problema, três foram resolvidos de forma ótima e para outras duas, utilizando arredondamento, soluções aproximadas foram encontradas. O tempo de processamento para estas variou de 658 a 1060 segundos.

É importante ressaltar que o algoritmo Grötschel e Win não é polinomial, nem exato; entretanto produziu soluções ótimas na maioria dos casos estudados.

3.2.3 Solução de Sherafat

Sherafat [She88] apresenta uma abordagem, cuja idéia básica é resolver o problema descrito por (1), (2), (3), (4), (5), (6) e (7) como um problema de circulação de custo mínimo. Para tanto, as restrições (2), as quais são intoleráveis pelos algoritmos de fluxo, são relaxadas e outras assimiláveis substituídas em seu lugar.

A restrição (3) garante que cada arco (v_i, v_j) apareça pelo menos uma vez no circuito euleriano, enquanto a restrição (2) garante o mesmo para cada aresta. Se x_{ij} for interpretada como fluxo em cada arco e aresta, estas duas restrições seriam aquelas de estabelecerem o limite inferior de fluxo em cada link. A formulação acima poderia ser interpretada, então, como uma formulação de circulação de custo mínimo, contanto que a restrição (2) fosse mais específica; ela asseguraria uma unidade de fluxo na aresta (v_i, v_j), sem no entanto especificar a orientação do fluxo.

O autor sugere uma transformação no grafo, substituindo cada aresta $e_p = (v_i, v_j)$ por uma estrutura denominada de *pseudo-aresta*, composta por quatro nós e cinco arcos, como mostrada na figura 3.1.

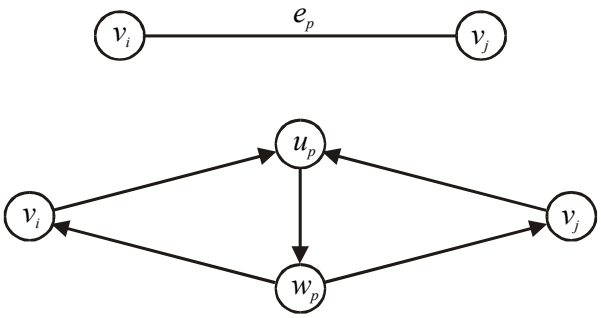


Figura 3.1

O arco (u_p, w_p) da pseudo-aresta terá o limite inferior de fluxo igual a 1, e seu custo será o custo original da aresta (v_i, v_j). Os demais arcos da estrutura terão custo igual a zero e o limite inferior de fluxo, também igual a zero. Todos os arcos da pseudo-aresta terão capacidade infinita de fluxo. Os arcos originais não passam por nenhuma alteração, e terão o seu custo original preservados, o limite inferior de fluxo igual a 1, e a capacidade infinita.

O grafo transformado desse modo será completamente orientado, com limites de fluxo estabelecidos em todos os arcos. O método "Out-of-kilter" pode ser usado para achar uma circulação de custo mínimo nesse grafo. A solução de circulação é uma tentativa de decidir, de maneira ótima, a orientação de fluxo em cada aresta (agora representada por uma estrutura orientada). Nessa solução, pelo menos uma unidade de fluxo estará circulando em cada uma dessas estruturas. Se a circulação de custo mínimo apresentar solução viável de fluxo para todas as pseudo-arestas, a solução ótima de PCCM é encontrada.

Numa solução de circulação, a passagem de fluxo numa pseudo-aresta (v_i, v_j) poderá ser de v_i a v_j, de v_j a v_i, ou, eventualmente, em ambos os sentidos. Entretanto, poderá acontecer uma situação não desejável – a *circulação triangular*. Isto é a situação em que o fluxo circula apenas num triangulo da estrutura. Quando isso acontece para alguma pseudo-aresta, não se tem a solução viável de fluxo para a respectiva aresta. Nesta condição se recorre a um procedimento de Branch-and-Bound, no qual uma pseudo-aresta com fluxo inviável é orientada num dos sentidos (fixando limites inferiores de fluxo apropriadamente nos demais arcos da estrutura).

O autor demonstra a eficiência do procedimento da busca binária, uma vez que o número de pseudo-arestas com fluxo inviável tende a se esgotar rapidamente, pela própria natureza dessas estruturas. O procedimento de busca não seria eficiente, se cada aresta, em vez de pseudo-aresta, fosse substituída simplesmente por um par de arcos contrariamente orientados. De fato, quando a orientação de fluxo é forçada num dos sentidos de uma pseudo-aresta, a unidade de fluxo tem que achar um circuito completo, de custo mínimo, para manter a viabilidade da circulação. O método Out-of-kilter, enquanto possível, desestimula a passagem de mais de uma unidade de fluxo nas pseudo-arestas já orientadas. Por isso, a orientação de uma destas pseudo-arestas obriga normalmente a orientação de

um número maior de pseudo-arestas na mesma situação, tornando viável o fluxo em todas elas, com simples reorientação de fluxo, e sem que isso acarrete em acréscimo de custo.

Os testes computacionais para este método foram feitos com 105 grafos aleatoriamente gerados, na faixa de $10 \leq |N| \leq 100$, $20 \leq |E| \leq 100$, e $20 \leq |A| \leq 150$. Soluções exatas foram encontradas em todos os casos tentados. Para os grafos maiores, o tempo de processamento foi de 54 segundos, em média, num computador IBM 4341.

Um fato relevante relatado pelo autor é a proximidade da primeira solução viável encontrada no processo da busca binária, com a solução exata encontrada no final. Em 30% dos casos esta primeira solução coincidiu com a otimalidade. Em média, a diferença foi em torno de 1% para todos os casos gerados. Por esta razão, o autor sugere também a utilização do método na forma de uma α-aproximação a partir do limite inferior.

3.2.4 Solução de Christofides et al.

Outra formulação de programação linear inteira foi proposta por Christofides et al. [Cri84]. Sejam $A_k^+ = \{(v_i, v_j) \in A : v_i = v_k\}$, $A_k^- = \{(v_i, v_j) \in A : v_j = v_k\}$ e N_k o conjunto de todos os nós ligados a v_k por uma aresta. Seja também x_{ij} o número adicional das vezes que o arco (v_i, v_j) é utilizado na solução ótima, e y_{ij} o número total das vezes que a aresta (v_i, v_j) é percorrido de v_i para v_j. Considere p_k uma constante binária igual a 1 se, e somente se, o grau do nó v_k é ímpar e z_k uma variável inteira. Então, a formulação é a seguinte:

Minimizar $$\sum_{(vi,vj)\in A} c_{ij}(1 + x_{ij}) + \sum_{(vi,vj)\in E} c_{ij}(y_{ij} + y_{ji}) \qquad (11)$$

Sujeito a

$$\sum_{(vi,vj)\in A_k^+} (1 + x_{ij}) - \sum_{(vi,vj)\in A_k^-} (1 + x_{ij}) + \sum_{vj\in N_k} y_{kj} - \sum_{vj\in N_k} y_{jk} = 0 \qquad (v_k \in N) \qquad (12)$$

$$\sum_{(vi,vj)\in A_k^+} x_{ij} + \sum_{(vi,vj)\in A_k^-} x_{ij} + \sum_{vj\in N_k} (y_{kj} + y_{jk} - 1) = 2z_k + p_k \qquad (v_k \in N) \qquad (13)$$

$$y_{ij} + y_{ji} \geq 1 \qquad ((v_i + v_j) \in E) \qquad (14)$$

$$z_k, x_{ij}, y_{ij}, y_{ji} \geq 0 \text{ e inteiros} \qquad (15)$$

O problema é resolvido por intermédio de um algoritmo enumerativo, no qual dois diferentes limites inferiores são calculados em cada nó da árvore de pesquisa. O primeiro limite é obtido pela relaxação da restrição (12) numa maneira lagrangeana, seguida da solução de um emparelhamento perfeito de custo mínimo. O segundo é obtido pela Relaxação Lagrangeana da restrição (14) e resolvendo um problema de fluxo de custo mínimo.

Utilizando este procedimento, os autores resolveram de forma exata 34 problemas aleatoriamente gerados, com $7 \leq |N| \leq 50$, $4 \leq |E| \leq 39$, e $3 \leq |A| \leq 85$. O tempo de CPU máximo observado ficou em torno de 500 segundos num UNIVAC 1100/60.

3.2.5 Solução de Nobert e Picard

Uma abordagem diferente foi proposta por Nobert e Picard [Nob96]. Na sua formulação existe apenas uma única variável associada a cada aresta, ao contrário das abordagens descritas acima que associam duas variáveis a cada uma. Portanto a solução de programação linear inteira não especifica uma orientação para as arestas. As restrições impostas asseguram que o grafo ampliado satisfaça as condições necessárias e suficientes de Ford e Fulkerson para unicursalidade. Isto é, garantem que o grafo seja par e balanceado.

Para apresentar a formulação, para cada subconjunto próprio S de N, definem-se os conjuntos

$$A^+(S) = \{(v_i, v_j) \in A : v_i \in S, v_j \in N \setminus S\},$$

$$A^-(S) = \{(v_i, v_j) \in A : v_i \in N \setminus S, v_j \in S\},$$

$$E(S) = \{(v_i, v_j) \in E : v_i \in S, v_j \in N \setminus S, \text{ ou } v_i \in N \setminus S, v_j \in S\},$$

e considerem-se $u(S) = |A^+(S)| - |A^-(S)| - |E(S)|$. Desta forma, se $S = \{v_k\}$, então $A^+(S) = A_k^+$, $A^-(S) = A_k^-$ e $E(S) = E_k$. Constantes p_k e variáveis z_k são definidas como acima, e apenas uma variável y_{ij} agora representa o número de cópias da aresta (v_i, v_j) a ser adicionado ao grafo para torná-lo euleriano. A formulação é dada abaixo:

$$\text{Minimizar} \quad \sum_{(v_i,v_j)\in A} c_{ij}x_{ij} + \sum_{(v_i,v_j)\in E} c_{ij}y_{ij} \qquad (16)$$

Sujeito a

$$\sum_{(v_i, v_j) \in A} x_{ij} + \sum_{(v_i,v_j) \in E} y_{ij} = 2z_k + p_k \qquad (v_k \in N) \qquad (17)$$

$$-\sum_{(v_i,v_j) \in A^+(S)} x_{ij} + \sum_{(v_i,v_j) \in A^-(S)} x_{ij} + \sum_{(v_i,v_j) \in E(S)} y_{ij} \geq u(S) \qquad (S \subset N, S \neq \phi) \qquad (18)$$

$$z_k, x_{ij}, y_{ij} \geq 0 \text{ e inteiros} \qquad (19)$$

Nessa formulação, as restrições (18) obrigam todos os subconjuntos próprios S de N a serem balanceados. Isso é feito pela imposição de um número suficiente de arcos e arestas introduzidos para compensar o desbalanceamento $u(S)$. Além dessas restrições, os autores introduzem uma forma generalizada das *desigualdades florais* (*blossom inequalities*):

$$\sum_{(v_i,v_j) \in A^+(S)} x_{ij} + \sum_{(v_i,v_j) \in A^-(S)} x_{ij} + \sum_{(v_i,v_j) \in E(S)} y_{ij} \geq 1 \qquad (S \subset N, S \text{ } \textit{ímpar}) \qquad (20)$$

Essas restrições são redundantes, mas ajudam a reforçar a relaxação a ser feita pelo procedimento. Inicialmente, o programa inclui todas as restrições de não-negatividade, condições de conjuntos balanceados (18) correspondentes aos nós desbalanceados e a maioria de conjuntos desbalanceados S de G, e as desigualdades florais generalizadas (20) associadas aos nós ímpares. O trabalho de Nobert e Picard inclui a descrição de um procedimento para identificar o subconjunto mais desbalanceado de G.

No decorrer do algoritmo, restrições adicionais de conjuntos balanceados e de desigualdades florais generalizadas são geradas, na medida que a sua violação é detectada. Quando isso não é mais possível, um número de cortes de Gomory é acrescentado a fim de atingir a integralidade no resultado. O procedimento pode terminar numa solução inteira, ou não-inteira. Se a solução for inteira e satisfaz todas as restrições, um grafo euleriano de custo mínimo é identificado. Caso contrário, um processo de branching deve se iniciar.

O algoritmo foi aplicado a um grande número de PCCM's aleatoriamente gerados, com $16 \leq |N| \leq 225$, $15 \leq |E| \leq 4455$, e $2 \leq |A| \leq 5569$. Dos 440 problemas gerados, 313 foram resolvidos em otimalidade, sem branching. O numero de restrições geradas no decorrer do algoritmo eram normalmente na ordem de $|N|$. Disso se pode concluir que o número de links não é um fator limitante para o algoritmo, mas sim, o número de nós.

Os maiores grafos do tipo pseudo-manhattan tentados tinham $|N|$ =169 e, em média, $|E|$ = 434, e $|A|$ = 136. O tempo médio de processamento para os casos resolvidos dessa dimensão foi de 139 segundos em um computador CDC Cyber 855. Dos 10 problemas desse porte gerados, apenas 4 foram resolvidos com sucesso; para os outros 6 a tentativa foi abandonada, decorridos 500 segundos de processamento. Tratando-se de grafos gerais, os 10 maiores tinham $|N|$ =80 e, em média, $|E|$ = 1363, e $|A|$ = 1165, dos quais 4 foram resolvidos com sucesso, com tempo de processamento médio de 307 segundos. Os autores relatam ainda a solução de um caso com $|N|$ =225, $|E|$ = 633, e $|A|$ = 341. Para este problema não foi relatado o tempo de processamento.

3.2.6 Abordagem de Minieka

Minieka [Min79] tem sugerido que PCCM pode ser formulado e resolvido como um problema de fluxo com ganhos. A chave de sua idéia é a prova da seguinte proposição:

Proposição - Em qualquer solução ótima de PCCM, se uma aresta é percorrida em ambas as orientações, então a aresta é percorrida exatamente duas vezes.

Isso implica que na solução ótima de PCCM apenas uma das seguintes situações pode ocorrer com cada aresta (x, y): a aresta é percorrida:

a) apenas no sentido de x a y;
b) apenas no sentido de y a x;
c) exatamente uma vez em cada sentido.

Uma vez tomada essa decisão para cada aresta, o número ótimo das vezes que cada arco e aresta deve ser repetida será encontrado como solução de um problema de fluxo de custo mínimo, num grafo completamente orientado. Para decidir sobre a orientação das arestas na solução ótima, Minieka sugere uma transformação no grafo em que cada aresta é substituída por uma estrutura composta por 4 nós e sete arcos. Por intermédio de um procedimento, a cada nó é atribuído um número que representa uma demanda (as demandas negativas são consideradas ofertas). No grafo resultante é resolvido um problema de fluxo com ganhos, conforme a demanda calculada em cada nó.

O autor reconhece a inexistência de um método eficiente para resolver a última instância do método proposto, a de achar uma solução inteira para o problema de fluxo

com ganho. Os algoritmos existentes para este problema encontram soluções fracionárias para o caso (Christofides [Cri75]). Ele sugere o emprego de alguma técnica de programação linear inteira para o desafio, entretanto, não apresenta uma formulação, nem relata alguma experiência computacional.

3.3 Soluções Heurísticas

3.3.1 Origens das Heurísticas para o PCCM

Vários autores têm sugerido algoritmos heurísticos para o Problema de Carteiro Chinês Misto. Esses algoritmos procuram uma boa solução, satisfazendo as condições necessárias e suficientes de unicursalidade. Uma estratégia bem explorada é a de acrescentar cópias dos links ao grafo, de modo a torná-lo par e simétrico. Como foi visto, estes dois requisitos juntos formam uma condição suficiente, porém não necessária, para a unicursalidade.

Edmonds e Johnson [Edm73] apresentaram um algoritmo para resolver o PCCM, para o caso particular em que o grafo originalmente é par, porém não é simétrico. O grafo é tornado simétrico, resolvendo um problema de fluxo em um grafo transformado. Este procedimento sugere uma heurística de duas fases para um grafo misto qualquer. Na primeira fase o grafo é transformado em par – isso pode ser feito resolvendo um problema de emparelhamento entre os nós de grau ímpar, ignorando a orientação dos arcos. Na segunda fase aplica-se o procedimento de fluxo sugerido por Edmonds e Johnson, tornando o grafo simétrico.

Frederickson [Fre79] demonstrou que este procedimento pode não funcionar, porque o grafo simétrico obtido na segunda fase pode não manter a paridade. Ele modificou o algoritmo, acrescentando uma terceira fase em que o grafo simétrico volta a recuperar a condição de paridade. Este algoritmo, referenciado na literatura como *Mixed 1*, está detalhado a seguir:

3.3.2 Algoritmo Mixed 1

Fase I – Conversão de G em um grafo par.

P1) Seja G^* o grafo obtido a partir de G, ignorando a orientação de todos os arcos.

P2) Resolva o PCC no grafo G^* usando o algoritmo de emparelhamento perfeito de Edmonds e Johnson [Edm73]. Sejam $Z(E)$ o conjunto de arestas e $Z(A)$ o conjunto de arcos obtidos pela solução de emparelhamento a serem acrescidos ao grafo. Sejam, também, $E_1=E\cup Z(E)$ e $A_1=A\cup Z(A)$. Portanto, $G_1=(N, E_1, A_1)$ é um grafo par.

Fase II – Transformação de G_1 num grafo simétrico.

P1) Construa um novo grafo $G_2=(N, A_2)$ com custos, capacidades e demandas definidos como abaixo:

- Para cada aresta $(i, j)\in E_1$, criar quatro novos arcos em A_2 conforme especificação a seguir:
  a) uma cópia de (i, j) com custo d_{ij} e capacidade infinita de fluxo;
  b) uma cópia de (j, i) com custo d_{ij} e capacidade infinita de fluxo;
  c) uma cópia de (i, j), denominada como (i, j)', com custo zero e capacidade unitária de fluxo;
  d) uma cópia de (j, i), denominada como (j, i)', com custo zero e capacidade unitária de fluxo;
- Para cada arco $(k, l)\in A_1$, cria uma cópia de (k, l) em A_2, com custo d_{kl} e capacidade infinita de fluxo.
- Para cada nó $i\in N$ defina:
  a) Demanda = $\{g_s(i)$ em $G_2\}-\{g_e(i)$ em $G_2\}$ se $g_s(i)$ em $G_2 > g_e(i)$ em G_2;
  b) Suprimento = $\{g_e(i)$ em $G_2\}-\{g_s(i)$ em $G_2\}$ se $g_e(i)$ em $G_2 > g_s(i)$ em G_2, onde $g_s(i)$ e $g_e(i)$ são, respectivamente, o grau de saída e o grau de entrada do nó i.

P2) Encontre o fluxo de custo mínimo no grafo G_2. Sejam Y_{ij}, Y_{ji}, $Y_{ij'}$, $Y_{ji'}$, e Y_{kl}, as quantidades de fluxo nos arcos (i, j), (j, i), (i, j)', (j, i)', e (k, l), respectivamente.

P3) Construa um grafo simétrico $G_3 = (N, E_3, A_3)$, conforme os seguintes passos. Inicialmente, faça $E_3 = \varnothing$ e $A_3 = A_1$.

a) se $Y_{ij'} + Y_{ji'} = 1$, coloque $Y_{ij'}$ cópias do arco (i, j) e $Y_{ji'}$ cópias do arco (j, i) em A_3;

b) se $Y_{ij'} + Y_{ji'} \neq 1$, coloque uma cópia da aresta (i, j) em E_3;

c) coloque Y_{ij} cópias do arco (i, j) e Y_{ji} cópias do arco (j, i) em A_3;

d) coloque Y_{kl} cópias do arco (k, l) em A_3.

Fase III – Recuperação da paridade de G.

Sejam A' o conjunto de arcos artificiais, gerado na fase II, e $E_4 = E_3$. Identifique os ciclos em $A' \cup E_4$, que consistem de caminhos alternados em A' e E_4, ancorados em cada extremidade por um nó de grau ímpar de G_3. Para encontrar tais ciclos, a orientação de arcos nos caminhos em A' deve ser ignorada. Como os ciclos cobrem todos os nós ímpares de G_3, orientações arbitrárias devem ser atribuídas aos ciclos. Os arcos nos ciclos serão ou duplicados ou deletados, dependendo se a orientação do ciclo é a mesma da orientação original dos arcos em A', ou não. As arestas serão orientadas de acordo com a orientação do ciclo. Conseqüentemente, todos os nós do grafo resultante serão pares, e o grafo continuará simétrico.

Frederickson [Fre79] mostrou que a complexidade computacional do algoritmo é $O(\max\{|N|^3,\ |A|(\max\{|A|,\ |E|\})^2)\}$. Ele mostrou, ainda, que o algoritmo é uma 1-aproximação, o que significa dizer que no pior caso, (solução de Mixed 1) / (solução ótima) ≤ 2, sendo possível atingir este limite.

Considere o exemplo da figura 3.2 (a), com seis nós, seis arestas e dois arcos. Os custos associados aos links estão especificados nessa figura, onde $M > \varepsilon > 0$. É fácil verificar que o algoritmo Mixed 1 irá produzir a solução mostrada na figura 3.2 (b), com custo total de $8M + 8\varepsilon$, enquanto a solução ótima tem um custo total de $4M + 10\varepsilon$. Portanto, tem-se que:

(solução de Mixed 1) / (solução ótima) = $(8M + 8\varepsilon) / (4M + 10\varepsilon) \rightarrow 2$, quando $M >> \varepsilon$

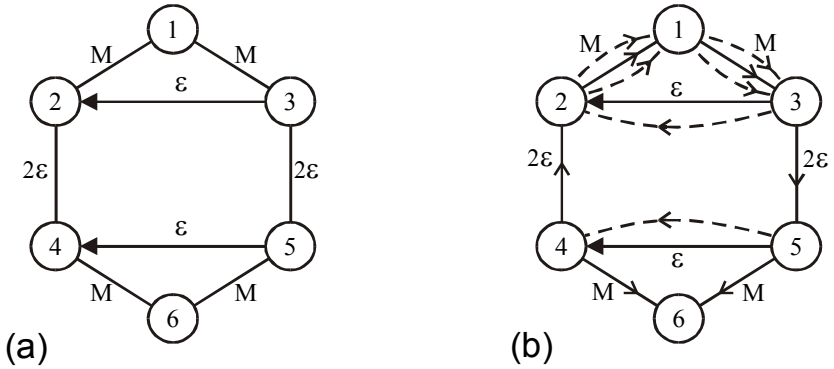


Figura 3.2

3.3.3 Algoritmo Mixed 2

Frederickson [Fre79], sugeriu, ainda, um outro procedimento alternativo, o qual é essencialmente o reverso do procedimento de Edmonds e Johnson [Edm73]. O método reverso faz um grafo simétrico na primeira etapa, tornando-o par na seqüência. Este algoritmo é referenciado na literatura como *Mixed 2*, e encontra-se descrito a seguir:

Fase I – Conversão de G em um grafo simétrico.

P1) Construa um novo grafo $G_1=(N, A_1)$ com custos, capacidades e demandas definidos como abaixo:

- Para cada aresta $(i, j)\in E$, criar quatro novos arcos em A_1 conforme especificação a seguir:
  a) uma cópia de (i, j) com custo d_{ij} e capacidade infinita de fluxo;
  b) uma cópia de (j, i) com custo d_{ij} e capacidade infinita de fluxo;
  c) uma cópia de (i, j), denominada como (i, j)', com custo zero e capacidade unitária de fluxo;
  d) uma cópia de (j, i), denominada como (j, i)', com custo zero e capacidade unitária de fluxo.
- Para cada arco $(k, l)\in A$, criar uma cópia de (k, l) em A_1, com custo d_{kl} e capacidade infinita de fluxo.
- Para cada nó $i\in N$ defina:
  a) Demanda = $\{g_s(i)$ em $G_1\}-\{g_e(i)$ em $G_1\}$ se $g_s(i)$ em $G_1 > g_e(i)$ em G_1;
  b) Suprimento = $\{g_e(i)$ em $G_1\}-\{g_s(i)$ em $G_1\}$ se $g_e(i)$ em $G_1 > g_s(i)$ em G_1, onde $g_s(i)$ e $g_e(i)$ são, respectivamente, o grau de saída e o grau de entrada do nó i.

P2) Encontre o fluxo de custo mínimo no grafo G_1. Sejam Y_{ij}, Y_{ji}, $Y_{ij'}$, $Y_{ji'}$, e Y_{kl}, as quantidades de fluxo nos arcos (i, j), (j, i), (i, j)', (j, i)', e (k, l), respectivamente.

P3) Construa um grafo simétrico $G_2 = (N, E_2, A_2)$, conforme os seguintes passos: Inicialmente, faça $E_2 = \varnothing$ e $A_2 = A$.

a) se $Y_{ij'} + Y_{ji'} = 1$, coloque $Y_{ij'}$ cópias do arco (i, j) e $Y_{ji'}$ cópias do arco (j, i) em A_2;
b) se $Y_{ij'} + Y_{ji'} \neq 1$, coloque uma cópia da aresta (i, j) em E_2;
c) coloque Y_{ij} cópias do arco (i, j) e Y_{ji} cópias do arco (j, i) em A_2;

d) coloque Y_{kl} cópias do arco (k, l) em A_2.

Fase II – Transformação de G_2 em um grafo euleriano.

P1) Resolva o problema do carteiro chinês no subgrafo de G, formado apenas de E_2, e os nós associados, usando o procedimento de emparelhamento de Edmonds e Johnson [Edm73].

P2) Sejam $Z(E_2)$ o conjunto de arestas obtido pela solução de emparelhamento, $E_3=E_2 \cup Z(E_2)$ e $A_3=A_2$. O grafo euleriano $G_3=(N, E_3, A_3)$ é a solução desejada.

Claramente a complexidade computacional de Mixed 2 é a mesma de Mixed 1, isto é, $O(\max\{|N|^3, |A|(\max\{|A|, |E|\})^2)\}$. O limite de seu pior caso, também, é a mesma de Mixed 1, e igualmente atingível. Para ilustrar este último ponto, considera-se o grafo da figura 3.3 (a), com oito nós, seis arestas e seis arcos. Os custos associados aos links estão especificados nessa figura, onde $M > \varepsilon > 0$. Obviamente o grafo é simétrico, portanto pode-se proceder diretamente com a fase II do algoritmo. É fácil verificar que o algoritmo Mixed 2 irá produzir a solução mostrada na figura 3.3 (b), com custo total de $12M + 9\varepsilon$, enquanto a solução ótima tem um custo total de $6M + 12\varepsilon$. Portanto, tem-se que:

(solução de Mixed 2) / (solução ótima) = $(12M + 9\varepsilon) / (6M + 12\varepsilon) \rightarrow 2$, quando $M >> \varepsilon$

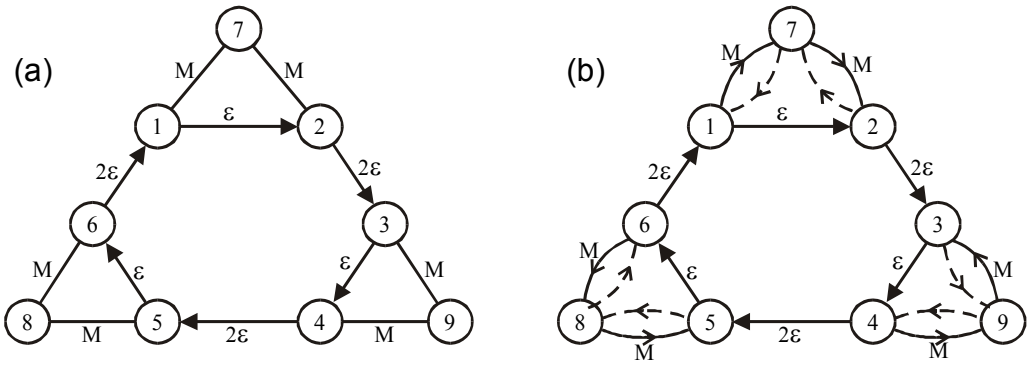


Figura 3.3

Para melhorar o pior caso, Frederickson [Fre79] considerou uma estratégia mista. O procedimento consiste em executar ambos os procedimentos, Mixed 1 e Mixed 2, para gerarem duas soluções completas de PCCM, e depois escolher a melhor entre as duas soluções. Ele mostrou que isso reduz a razão entre o pior caso e a solução ótima para $^5/_3$.

De fato, se a estratégia mista fosse aplicada nos últimos dois exemplos, em ambos a solução ótima seria encontrada.

3.3.4 Comparação dos Algoritmos Mixed 1 e Mixed 2

Frederickson não apresentou testes computacionais para os algoritmos acima. Entretanto Pearn e Liu [Pea95.1] testaram ambos os algoritmos com um grupo de 60 problemas aleatoriamente gerados. Os resultados indicaram que:

i) Mixed 1 tem um desempenho significativamente superior nos grafos que tem de 0 a 70% de seus links orientados;
ii) Mixed 2 tem um desempenho significativamente superior nos grafos que tem de 70 a 100% de seus links orientados;
iii) Na maioria dos grafos mistos testados com poucos links orientados, o algoritmo Mixed 1 encontra a solução ótima.

Para se ter uma idéia da qualidade das soluções, os resultados foram comparados com os limites inferiores calculadas para cada caso. Os limites inferiores foram estabelecidos, resolvendo o PCC e ignorando a orientação dos arcos. Com isso, as soluções de Mixed 1 ficaram com o desvio médio de 33% em relação aos respectivos limites inferiores para grafos com maior percentual de arcos, diminuindo este número para abaixo de 1% para os casos de menor percentual de arcos. Para o Mixed 2, estes desvios médios ficaram entre 27% e 4% respectivamente. Os piores desvios registrados foram de 86% para Mixed 1 e 61% para o Mixed 2.

Os testes efetuados por Pearn e Liu se limitaram a grafos relativamente pequenos, com $10 \leq |N| \leq 35$, $2 \leq |E| \leq 266$, e $3 \leq |A| \leq 302$. O tempo de processamento nunca passou de 2 segundos, num computador PC-486. Não houve diferença significativa de tempo entre um algoritmo e o outro.

3.3.5 Algoritmo de Christofides et al.

Christofides et al. [Cri84], além de um método exato, anteriormente discutido, apresentaram um algoritmo aproximado que é equivalente ao Mixed 2 descrito acima, e não será repetido nesse espaço. Entretanto, vale ressaltar os resultados computacionais por eles alcançados, os quais foram comparados às soluções ótimas. Foram testados 34

problemas aleatoriamente gerados, na faixa de $7 \leq |N| \leq 50$, $4 \leq |E| \leq 39$, e $3 \leq |A| \leq 85$. A heurística produziu soluções com valores em média 3% acima da ótima. O pior desvio foi de 17%.

3.3.6 Algoritmos Mixed 1 e Mixed 2 Modificados

Pearn e Liu [Pea95.1] observaram que os algoritmos Mixed 1 e Mixed 2, apresentados por Frederickson [Fre79], podem ser melhorados com implementação de estratégias adicionais. O algoritmo Mixed 1, por exemplo, aplica inicialmente uma rotina de emparelhamento e depois resolve um problema de fluxo. Ambos os procedimentos adicionam cópias extras de arcos e arestas ao grafo. Eles detectaram que não são raras as situações em que tais cópias formam circuitos (principalmente quando o grafo contém um percentual maior de arcos). Obviamente, se isso acontecer, todo o circuito formado por cópias extras pode ser removido, sem que isso altere a paridade ou a simetria do grafo.

O *algoritmo Mixed 1 modificado* contém todas as três fases do Mixed 1, mais uma quarta fase em que os circuitos completamente formados com arcos e arestas artificiais são identificados e removidos do grafo. Como esta última fase tem a complexidade de $O(|N|^2)$, então a complexidade de Mixed 1 modificado permanece a mesma de Mixed 1.

O *algoritmo Mixed 2 modificado* é o mesmo de Mixed 2, exceto na fase II, quando é calculado o emparelhamento mínimo. Os caminhos mínimos entre pares de nós de grau ímpar são calculados no grafo original (misto), ignorando as orientações de todos os arcos. Em seguida, para cada par de nós na solução de emparelhamento, o caminho mínimo que une o par (que pode ser orientado nesse caso) é acrescentado ao grafo. Se o caminho acrescentado é orientado, obviamente, em contrapartida, algumas arestas do grafo devem ser orientadas, a fim de manter a simetria do grafo resultante.

Os autores testaram os dois algoritmos modificados com o mesmo conjunto de 60 grafos usados nos testes de Mixed 1 e Mixed 2. Os testes computacionais mostraram os seguintes resultados:

i) O algoritmo Mixed 1 modificado superou o Mixed 1, e a melhoria variou de 0,0 a 4,4%, sendo mais acentuada, quanto maior o número de links orientados (arcos);

ii) O algoritmo Mixed 2 modificado superou o Mixed 2, e a melhoria variou de 1,1 a 3,4%, sendo mais acentuada quanto maior o número de links não-orientados (arestas);
iii) Mixed 1 modificado superou Mixed 2 modificado para grafos com maior percentual de arestas;
iv) Mixed 2 modificado superou Mixed 1 modificado para grafos com maior percentual de arcos.

3.3.7 Algoritmos de Mixed 1 e Mixed 2 Aperfeiçoados

Pearn e Chou [Pea99] proveram melhoramentos adicionais para os algoritmos Mixed 1 e Mixed 2 modificados, a partir de um refinamento final do grafo ampliado. A estratégia é a mesma para os dois algoritmos.

Seja $G^* = (N, A^*)$ o grafo ampliado obtido pelo algoritmo Mixed 1 modificado (ou Mixed 2 modificado), e seja $A^+ \subset A^*$ o conjunto de todos os arcos artificiais que foram acrescentados a *G* para obter o grafo ampliado G^*. O refinamento consiste na execução do seguinte procedimento:

i) encontre a solução de fluxo de custo mínimo no grafo $G_1 = (N, A^* - A^+)$ (o qual é completamente orientado, obtido do grafo G^* com remoção de todos os arcos artificiais A^+);
ii) seja $A^\#$ a solução de fluxo. Substitua A^+ por $A^\#$ para obter uma nova solução $G_2 = (N, (A^* - A^+) \cup A^\#)$.
iii) para cada aresta que tem duas orientações em G^*, cada orientação deve ser escolhida para gerar uma solução completa, e a melhor das duas é selecionada.

Desde que todos os arcos artificiais (A^+) na solução de Mixed 1 modificado são removidos e substituídos por uma solução de fluxo de custo mínimo ($A^\#$), seria obvio afirmar que

- (solução de Mixed 1 aperfeiçoado) ≤ (solução de Mixed 1 modificado);
- (solução de Mixed 2 aperfeiçoado) ≤ (solução de Mixed 2 modificado).

Os testes computacionais foram feitos, usando a mesma família de grafos usada para testar as versões anteriores dos algoritmos Mixed. Os resultados comprovaram melhoria na qualidade de soluções geradas pelas versões aperfeiçoadas de Mixed 1 e Mixed 2, comparada com as obtidas por Mixed 1 e Mixed 2 modificados. As melhorias mais acentuadas aconteceram pelo Mixed 1 aperfeiçoado, principalmente nos piores casos.

3.3.8 Um Algoritmo de ½-Aproximação

Como foi visto, o pior caso dos algoritmos Mixed 1 e Mixed 2 originais é de duas vezes a respectiva solução ótima. Frederickson [Fre79] demonstrou que se for aplicada a estratégia mista (aplicar ambos os algoritmos e escolher a melhor solução), a razão entre a solução aproximada e a ótima cai para 5/3, no pior caso. Raghavachari e Veerasamy [Rag99] fizeram uma adaptação da estratégia mista, e garantiram a razão de 3/2 para o pior caso.

Os autores obtiveram um novo limite inferior para o custo de uma solução ótima, usando as propriedades de soluções intermediárias do PCCM, encontradas nos procedimentos de Mixed. A idéia do algoritmo é uma exploração apropriada desse novo limite inferior.

O algoritmo proposto aplica ambos os procedimento Mixed 1 e Mixed 2, porém efetua uma alteração dos custos, antes da aplicação de Mixed 1. Suponha que a rotina de balanceamento, utilizado nos algoritmos Mixed, seja aplicada a um grafo $G = (N, A, E)$, encontrando o grafo balanceado $G_b = (N, M, U)$. $U \subseteq E$ são arestas de G que não foram orientadas pelo procedimento, e $M \supseteq A$ os arcos que satisfazem o balanceamento $g_e = g_s$ em cada nó. Como passo inicial, este processo de balanceamento é aplicado, os conjuntos *M* e *U* são identificados, e o custo de todos os arcos e arestas que estão em *M* fixados a zero. Com esta mudança de custos, Mixed 1 é aplicado, forçando que apenas os links que pertencem a *M* sejam duplicados na fase de emparelhamento perfeito.

Mixed 2 é aplicado, sem nenhuma mudança, e naturalmente na sua fase de emparelhamento perfeito poderá duplicar algumas arestas que pertencerão apenas a *U*. Ao final, a melhor solução entre as duas obtidas será escolhida. É demonstrado que, com essa estratégia, um dos dois procedimentos de Mixed produzirá a solução com desvio máximo de 0,5 com relação ao limite inferior de custo obtido.

Os autores não apresentam testes computacionais para comparar o desempenho real do algoritmo com o original, e com as suas versões melhoradas [10, 21].

3.4 Considerações Sobre as Soluções Exatas e Heurísticas

As soluções exatas de PCCM relatadas na seção 3.2 mostram uma riqueza na variedade de abordagens empregadas pelos autores. Dentre elas, quatro estão acompanhadas com experiências computacionais, as quais estão fortemente limitadas pelo tamanho do problema. É difícil fazer uma distinção precisa do desempenho de um método em relação a outro, dado que os problemas testados e os computadores utilizados em cada caso são diferentes. No entanto, de um modo geral, a solução de Nobert e Picard [Nob96] parece ser a mais eficiente entre todas e, em se tratando de grafos densos, com certeza seria o método indicado. O mesmo não se pode dizer para o caso de grafos esparsos (com mais nós e menos links). De fato seus testes foram limitados a grafos gerais de até 80 nós, apresentando sucesso em 40% dos casos. Portanto, conclui-se que no estado atual da arte, as soluções exatas ainda são inadequadas para a maioria dos problemas reais.

Visto as dificuldades com relação às soluções exatas, em geral inerentes aos problemas NP-completos, o caminho natural de solução é via algoritmos heurísticos, ou meta-heurísticos, que possam fornecer uma solução de boa qualidade, em tempo hábil, aplicáveis a problemas de porte maior. As soluções aproximadas, relatadas na seção 3.3 mostram várias versões para os algoritmos básicos Mixed 1 e Mixed 2, com destacadas contribuições de Pearn [10, 21], Frederickson [Fre79], e Raghavachari e Veerasamy [Rag99]. Experiências computacionais são relatadas apenas por Pearn e mostram basicamente o desempenho relativo de diferentes versões de algoritmos de Mixed entre si, utilizando grafos pequenos. Nenhum relato é apresentado acerca de comportamento desses algoritmos diante de problemas de maior porte.

O que chama atenção, é que ao contrário dos métodos exatos, destacados pela variedade de abordagens, os métodos heurísticos se resumem a diferentes versões destas duas abordagens, que basicamente são uma só, pois apenas seus principais passos aparecem em ordens diferentes. Também, não se tem notícia de nenhuma meta-heurística desenvolvida para o caso. Conclui-se então pela necessidade de desenvolvimento de novas

soluções aproximadas para o PCCM, ou para casos mais genéricos que envolvem este como instância particular.

IV – Problema do Carteiro Rural

4.1 Introdução

Embora o PCCM, estudado na seção anterior tenha diversas aplicações nos problemas de logística e de distribuição de bens e serviços, a maioria dos problemas reais nessas áreas se formula como um Problema de Carteiro Rural – PCR. Na sua forma geral, o problema é definido num grafo misto fortemente conexo $G = (N, A, E)$. A cada link $l \in A \cup E$ é associado um custo não-negativo d_l. Um subconjunto de links $A' \subseteq A$ e $E' \subseteq E$ precisam ser servidos. Na forma clássica, o PCR é definido num grafo não orientado, porém como foi visto no capítulo 2, pode ser formulado também num grafo orientado – PCRO, ou num grafo misto – PCRM.

Ao contrário do PCC, o PCR é NP-hard, mesmo nos casos orientado e não-orientado. Entretanto, se o grafo *G*'=(*N*,*A*',*E*') for conexo (não necessariamente fortemente conexo), o Problema de Carteiro Rural em G pode ser formulado como um de Carteiro Chinês num grafo reduzido a partir de *G*, e portanto terá solução polinomial nos casos orientados e não-orientados. Nestes casos, o procedimento consiste em calcular o grau de cada nó no grafo *G'*, considerando apenas os links requeridos, e calcular os caminhos mínimos, utilizando o grafo *G* por inteiro.

4.2 Soluções para o PCR

A seguinte heurística foi sugerida por Eiselt et al. [Eis95.2] para o PCR. Considere um grafo não-orientado $G = (N, E)$, com $E' \subset E$ como conjunto requerido de arestas. Seja $V \subseteq N$ o conjunto de nós cobertos pelos arcos em E'. O algoritmo resolve o problema no grafo transformado $G' = (V, R)$, onde R é formado pelo conjunto de arestas E' acrescido por arestas que representam caminhos mínimos entre todos os nós em V. Se G' for formado apenas pelas arestas E', seria desconexo, tendo p componentes conexos $G_1, G_2, ..., G_P$ com respeito aos conjuntos de nós $V_1, V_2, ..., V_P$ que formariam uma partição de V.

Heurística de Eiselt et al. [Eis95.2]

1. Forme o grafo transformado $G'=(V,R)$, a partir do grafo não-orientado $G=(N,E)$, como descrito acima.
2. (Árvore Geradora Mínima)
   Construir uma árvore geradora mínima *T* em *G'*, conectando $G_1, G_2, ..., G_P$
3. (Emparelhamento Mínimo)
   Determine os nós de grau ímpar com relação às arestas $E' \cup T$. Calcule o emparelhamento *M* de custo mínimo, entre estes nós de grau ímpar.
4. (Ciclo Euleriano)
   Obtenha um ciclo euleriano no grafo $G_e=(V, E' \cup T \cup M)$, o qual é uma solução para o PCR.

É possível demonstrar que a heurística acima é uma 0,5-aproximação para o PCR. A solução gerada por este método eventualmente pode variar, mudando a raiz da árvore geradora. Isto possibilita a repetição do procedimento, tomando cada vez um nó diferente como raiz, e escolhendo a melhor solução entre todas.

Pearn e Wu [Pea95.2] apresentaram duas outras versões para o algoritmo de Christofides, denominados como o *algoritmo modificado* e o *algoritmo reverso*. Os testes computacionais foram realizados em 200 problemas de até 50 nós, 199 arestas, e 61 arestas requeridas. Ambas as versões propostas mostraram uma pequena melhoria em relação ao algoritmo original. Comparando entre si, o algoritmo reverso mostrou um desempenho melhor que o algoritmo modificado, apenas quando os nós são majoritariamente de grau impar.

Visando soluções exatas, duas formulações de programação linear inteira têm sido propostas para o PCR. A primeira, sugerida por Christofides et al. [Cri81] permite incorporar na função objetivo, sob a forma de relaxação lagrangeana, a restrição que garante a paridade do grau de cada nó. A solução ótima é obtida através de um procedimento de *branch-and-bound*. Vinte e quatro problemas de até 84 nós, gerados aleatoriamente, foram resolvidos de modo exato.

Uma formulação diferente foi sugerida por Corberán e Sanchis [Cor98]. Com estudo e descrição parcial do poliedro de soluções do PCR, sugerem a geração de um conjunto de

desigualdades válidas, incluindo facetas, para serem usadas num algoritmo de planos de corte, associado a uma rotina de *branch-and-bound*. O esquema foi testado para o mesmo conjunto de 24 problemas usados por Christofides et al. [Cri81]. Vinte e três destes problemas foram resolvidos otimamente, na raiz da árvore de busca (sem nenhum branching), usando apenas os planos de corte.

4.3 Problema de Carteiro Rural Orientado - PCRO

Para o caso orientado do PCR, onde $A' \subset A$ e $E = \varnothing$, Christofides, et. al. [Cri86] sugerem um procedimento semelhante ao do caso não-orientado, descrito acima. Os passos da heurística são modificados para atender a um grafo orientado: no passo 3, é resolvido um problema de transporte ao invés do problema de emparelhamento perfeito. Usando a estratégia de mudança de raiz da árvore geradora, os autores relatam soluções com desvios médios de 1.3% da otimalidade, com pior caso registrado em 5%.

Os autores sugerem, também, um procedimento exato para o PCRO, usando uma formulação de programação linear inteira, semelhante àquela usada para o caso não-orientado, contida no Christofides et al. [Cri81]. O algoritmo é baseado essencialmente num procedimento de *branch-and-bound*, usando relaxação lagrangeana. Os autores apresentam testes computacionais com grafos de até 80 nós, 180 arcos e 74 arcos requeridos. Dos vinte e quatro problemas testados, vinte e três foram resolvidos de forma ótima, usando este algoritmo.

Problema do Carteiro Rural Agrupado - PCRA

O problema de carteiro rural agrupado (Clustered Rural Postman Problem) é uma versão particular do PCRO. Nesse problema, o conjunto de arcos requeridos A' é constituído de k componentes fracamente conexos $A'_1, A'_2, ..., A'_k$, cada um denominado de um agrupamento, de modo que todos os arcos de um agrupamento devem ser servidos, antes do roteiro prosseguir para um outro agrupamento.

Dror e Langevin [Dro97] apresentam uma abordagem de solução para PCRA em que o problema é transformado em um *Problema de Caixeiro Viajante Generalizado – PCVG*. O objetivo para o PCVG é achar um circuito de custo mínimo que inclui exatamente um nó de cada um dos agrupamentos. Dentro de cada agrupamento, um PCCO é resolvido,

usando os métodos apresentados por Beltrami e Bodin [Bel74], ou Edmonds e Johnson [Edm73]. Finalmente, os circuitos de PCCO são aninhados ao circuito de PCVG.

Os autores relatam experiências computacionais, usando 31 grafos aleatoriamente gerados de até 81 nós e 247 arcos, dos quais até 81 requeridos, e de até 15 componentes. O tempo de processamento relatado para os problemas de 50 nós ficou entre 51 a 130 segundos, enquanto para o problema maior (de 81 nós) foi mais que 2700 segundos, num computador Sparcstation 2, de 28,5 mips.

4.4 Problema de Carteiro Rural Misto - PCRM

Este é o caso mais genérico do PCR e um dos casos mais genéricos entre todas as formulações dos problemas de roteamento de arcos. Nesse caso, o problema é definido para os seguintes conjuntos requeridos: $N' = \varnothing$, $A' \subset A$ e $E' \subset E$. Todos os casos do PCR estudados acima, e todos os casos do PCC estudados anteriormente podem ser formulados como casos particulares do PCRM. Como foi visto, a maioria destes casos (como PCCM, PCR, e PCRO) é NP-hard. Portanto pode-se concluir que o PCRM, sendo a generalização desses, é também NP-hard.

Corberán et al. [Cor00] apresentaram dois métodos heurísticos para o PCRM. O primeiro é um algoritmo construtivo que obtém uma boa solução num tempo computacional relativamente curto. O método é parcialmente baseado numa abordagem exata para solução do PCCM, proposta por Christofides et al [Cri84]. O segundo é um procedimento de busca local que produz soluções de boa qualidade num tempo razoavelmente maior.

Algoritmo Construtivo para o PCRM (Corberán et al. [Cor00])

Nesse método, a exemplo de outras abordagens para o PCR e com objetivo de simplificação, o grafo é inicializado por meio de uma transformação, pela qual ele passa a conter apenas os nós terminais dos links requeridos, e com acréscimo de alguns arcos que representam os caminhos mínimos no grafo original [Eis95.2], [Cri86]. Após isso, ao longo de quatro estágios, o algoritmo constrói um grafo euleriano contendo todos os links requeridos, e atribui orientação a cada aresta requerida. Os estágios estão sucintamente descritos a seguir:

Inicialmente é construída uma árvore geradora mínima que conecta todos os componentes do grafo simplificado, a qual é acrescentada ao grafo, a fim de torná-lo conexo; depois com a solução de um problema de fluxo, além de conexo, o grafo passa a ser balanceado. Em seguida todos os acréscimos de links são aperfeiçoados com a solução de um novo problema de fluxo. O grafo resultante, que continua sendo conexo e balanceado, pode ter nós de grau ímpar. Resolvendo um problema de emparelhamento de custo mínimo, é obtida uma solução viável para o PCRM. Na etapa final, todas as cópias acrescentadas e todos os links não requeridos cuja remoção não desconecta o grafo são removidos, e um novo problema de fluxo é resolvido. O grafo aumentado conforme esta última solução é balanceado, completamente orientado e contém todos os links originalmente requeridos.

Algoritmo de Busca Tabu para o PCRM (Corberán et al. [Cor00])

O segundo método apresentado por Corberán et al. é uma adaptação da técnica de busca tabu para a solução do PCRM. A *Busca Tabu* é uma meta-heurística que pode ser usada para guiar um procedimento de busca local para um ótimo global, usando uma estrutura de memória que evita as armadilhas de ótimos locais [Glo97].

O procedimento começa com uma rota do PCRM, obtida pelo algoritmo construtivo descrito acima. Como foi dito, esta solução é fornecida na forma de um grafo euleriano conexo, contendo todos os links originalmente requeridos. A partir desta, um procedimento de busca tabu alterna entre duas rotinas básicas denominadas de *Intensificação* e *Diversificação*. Na rotina de intensificação, um arco de conexão (não requerido) é escolhido aleatoriamente, com probabilidade proporcional ao seu custo, removido e substituído por um outro que conecta os mesmos componentes, via um caminho mínimo. O arco removido entra na lista tabu. A fase de intensificação termina, quando todos os arcos de conexão são tentados por uma substituição válida. Nessa fase a função objetivo decresce, ou mantém seu valor.

Na rotina de diversificação, em cada iteração um arco de conexão não-tabu é escolhido. Se existe uma substituição válida por um outro arco não-tabu, a alteração é feita. Caso contrário, admite-se a substituição do arco escolhido com piora no valor da função objetivo.

A alternância entre os dois movimentos acima produz um grande número de soluções viáveis. O procedimento continua até atingir um critério de parada, que em geral é o não melhoramento da solução num ciclo completo entre as alternâncias.

Testes computacionais com 270 grafos aleatórios foram feitos pelos autores, usando um computador pessoal equipado com processador Pentium 150 Mhz. As soluções geradas foram comparadas com bons limites inferiores estabelecidos por um procedimento de planos de corte baseado num modelo de Programação Linear. Ambos os métodos mostraram soluções de boa qualidade: a percentagem do desvio médio em relação ao limite inferior foi menor que 2,3% para o algoritmo construtivo, e menor que 0,3% para o algoritmo da busca tabu. Os grafos de maior porte usados nos testes foram de 100 nós com mais de 200 links requeridos. Para estes, enquanto o tempo de computação ficava em torno de 1 segundo para o método construtivo, o mesmo variava de 103 a 1038 segundos, dependendo do número de componentes, para o método da busca tabu.

Solução de Laporte [Lap97]

Laporte [Lap97] sugeriu a transformação do PCRM para um Problema de Caixeiro Viajante – PCV. De fato, ele apresenta um método de transformação que serve para uma instância mais genérica dos Problemas de Roteamento de Arcos, a qual inclui, além do PCRM, alguns outros casos.

O autor sugere uma transformação de três etapas no grafo original. Na primeira etapa, cada aresta e_{ij} é substituída por um par de arcos a_{ij} e a_{ji}, resultando num grafo completamente orientado $G_1 = (N, A_1)$.

A segunda etapa consiste em transformar o Problema de Roteamento de Arcos, em um problema equivalente de Roteamento de Nós, num grafo completo $G_2 = (W, B)$. Nesse grafo, W consiste em um conjunto de nós, cada um associado a um arco de A_1, e B é o conjunto de todos os arcos que conectam os pares de nós em W. Mais precisamente, se a_{ij} e a_{kl} são dois arcos de A_1, então se define o arco $b_{jk} \in B$, com custo igual ao comprimento do caminho mínimo de x_j a x_k em G_1. No final dessa etapa, o problema original de Roteamento de Arcos em G é transformado num *Problema de Caixeiro Viajante Generalizado – PCVG*, em G_2.

A terceira e última etapa consiste em transformar o PCVG num PCV, usando regras descritas por Noon e Bean [Noo93]. Portanto, depois de concluída a transformação, o trabalho é resolver um problema padrão de caixeiro viajante em um grafo assimétrico.

O autor experimentou duas rotinas para a solução do PCV: uma exata, desenvolvida por Carpaneto e Toth [68], e outra heurística, de Karp [Kar79]. Os testes computacionais para resolver o PCRM foram realizados em grafos aleatórios com $40 \le |N| \le 220$, $80 \le |A| \le 660$, $20 \le |A'| \le 495$, e com um pequeno número de arestas: $1 \le |E'| \le 10$. Usando a rotina exata de PCV, o tempo de processamento foi razoável apenas para os problemas pequenos. De fato, das 1100 instâncias do PCRM testadas, 157 não foram resolvidas com o este método, esgotado um tempo limite de 5000 segundos para o processamento, num computador Silicon Graphics - IRIX5.3. A maior dificuldade encontrada nos testes foi com relação aos grafos que têm um maior número de arestas. Para o método heurístico, o desvio médio em relação à solução ótima ficou em torno de 2%. Para estes, o tempo de processamento não foi relatado.

O autor atribui a deficiência computacional à degeneração na estrutura de custo do grafo, inerente à transformação proposta pelo método. Entretanto, o mérito da abordagem reside no fato de prover uma ferramenta única para resolver uma variedade de Problemas de Roteamento de Arcos.

Problema do Carteiro Rural Misto com Conversões Penalizadas - PCRMCP

Corberán et al. [Cor02] estudaram uma versão do PCRM que incorpora as restrições de conversões nos vértices. Com estas restrições é possível levar em conta algumas regras básicas de trânsito urbano que se referem a conversões proibidas nos cruzamentos. O circuito de carteiro deve passar por todos os links requeridos, sem que cometa alguma conversão proibida em qualquer um dos nós.

Os autores apresentam um método de transformação polinomial, na linha de Laporte et al. [Lap97], transformando o PCRMCP em um Problema de Caixeiro Viajante Assimétrico – PCVA. A transformação constitui-se na construção de um novo grafo orientado G', a partir do grafo original G. Cada arco requerido em G é representado por um nó em G', e cada aresta, por um par de nós. Os arcos em G' são caminhos mínimos em G, os quais levam em conta as penalizações devidas às passagens proibidas, ou indesejáveis nos vértices.

O PCVA resultante foi resolvido usando um algoritmo exato, e um heurístico. Como método exato foi usado o procedimento de *branch-and-bound* de Fischetti e Toth [Fis92], o qual é considerado um dos melhores já publicados. E como heurístico, foi escolhido o *Patching Algorithm* devido a Karp [Kar79].

Os autores relatam testes computacionais com 216 grafos aleatoriamente gerados, dimensionados nas faixas de $40 \le |N| \le 200$, $90 \le |A| \le 440$, $36 \le |A'| \le 440$, e $10 \le |E'| \le 40$. O método exato apresentou-se adequado apenas para problemas com poucas arestas requeridas. Dos 216 problemas testados, 57 não foram resolvidos, dentro de um tempo limite de 28800 segundos, num Pentium II – 350 MHz. O método heurístico, embora apresentando soluções para todas as instâncias, e em geral com tempo abaixo de 10 segundos, mostrou-se também sensível ao número de arestas requeridas, no que se refere à qualidade das soluções. Enquanto para os problemas compostos de 10 arestas requeridas a solução heurística ficou em média 3,4% acima da solução ótima, para os problemas com 40 arestas requeridas, esta defasagem aumentou para 11,2% acima da ótima, considerando os casos com a solução ótima conhecida.

Considerando o decréscimo de desempenho dos algoritmos baseados em transformação do PCRMCP em PCVA com o aumento de número de arestas requeridas, Corberán et al. [Cor02] apresentaram uma heurística, composta de três fases. Inicialmente várias soluções viáveis para o PCRM (isto é, sem levar em conta as restrições nos vértices) são geradas, usando o algoritmo Busca Tabu, descrito em Corberán et al [Cor00]. Depois estas soluções são modificadas no sentido de atender as restrições do PCRMCP. Finalmente, numa terceira fase, alguns procedimentos de melhoramento são aplicados para refinar o circuito final.

Os testes computacionais com os mesmos grafos acima mencionados mostraram que este método não tem uma sensibilidade especial com relação ao número de arestas. Para os casos com a solução ótima conhecida, o método produziu soluções com desvio médio de 1% em relação à solução ótima. O tempo médio de processamento entre todas as instâncias foi de 12 segundos, e o máximo registrado para uma instância particular foi de 298 segundos.

4.5 Problema de Carteiro Rural com Vento - PCRV

O PCRV tem uma formulação semelhante a PCR, com $N' = \varnothing$, $A = \varnothing$ e $E' \subseteq E$, porém com um grau de liberdade a mais: as arestas, requeridas ou não, podem ser percorridas em cada sentido com custos desiguais. Com definição apropriada dos custos para arestas em cada um de seus sentidos, todas as instâncias do PCR estudadas acima, inclusive o PCRM, podem ser formuladas como casos particulares do PCRV. Por razões óbvias, este é também um problema NP-hard.

Por ter uma formulação simples e que ao mesmo tempo abrange quase a totalidade dos casos não-capacitados de Problemas de Roteamento de Arcos, o PCRV vem recebendo atenção especial em alguns trabalhos recém publicados. Entre estes podem ser destacados Benavent et al. [Ben03.1], e Benavent et al [Ben03.2].

Dado um grafo fortemente conexo $G = (N, E)$ com o conjunto de arestas requeridas $E' \subseteq E$, a solução do PCRV é um multi-grafo fortemente conexo $G^* = (N, A)$ que satisfaz (Benavent et al. [Ben03.1]):

- Cada arco $(i, j) \in A$ é uma cópia de uma aresta em *E*, com uma dada orientação;
- Para cada $(i, j) \in E'$, $(i, j) \in A$, ou $(j, i) \in A$;
- Cada nó em G^* é simétrico (isto é, seu grau de entrada é igual ao grau de saída).

Solução Exata de Benavent et al. [Ben03.1]

A idéia do método exato sugerido pelos autores é formular o PCRV como um problema de Programação Linear Inteira. Inicialmente, o problema é resolvido apenas utilizando a função objetivo com as restrições triviais, algumas outras que garantem a conectividade, e com a relaxação da condição de integridade. Um procedimento de Plano de Corte introduz gradativamente ao problema três famílias de desigualdades válidas, conhecidas como Cortes *R-Odd*, Desigualdades $K - C$, e Desigualdades *Honeycomb*. Estas desigualdades que descrevem o poliedro de soluções do PCRV, já estão conhecidas para algumas outras instâncias dos Problemas de Roteamento de Arcos [6, 29, 34, 35, 37]. Na medida que o algoritmo detecta a violação de qualquer uma destas desigualdades, ela é acrescida ao problema. O procedimento de corte continua até que nenhuma violação seja mais detectada. Se a solução obtida ainda não for inteira, uma rotina de *branch-and-bound* é invocada.

O trabalho apresenta testes computacionais com o conjunto de grafos aleatórios usados por Christofides et al. [Cri81], alterando apenas os custos das arestas, e com dois conjuntos de grafos obtidos a partir de redes de ruas de duas cidades da Espanha. Os grafos aleatórios são de dimensões menores, tendo até 84 nós, e até 74 arestas requeridas, enquanto os grafos obtidos a partir de malhas urbanas têm até 196 nós e 316 arestas, e até 70% das quais requeridas. Das 288 instâncias testadas, 185 foram resolvidas otimamente, usando apenas os planos de corte. As demais também foram resolvidas, recorrendo ao procedimento de *branch-and-bound*. Para os problemas de maior porte, o tempo médio de processamento foi de 62 segundos, e o máximo registrado em 109 segundos, num computador equipado com processador Pentium III - 1GHz.

Soluções Heurísticas de Benavent et al. [Ben03.1]

No mesmo trabalho, os autores apresentam 3 soluções heurísticas para o PCRV. Abaixo se encontra uma descrição sucinta da Heurística 1, a qual é baseada num trabalho de Zaw Win para o PCCV. Win [Win89] sugeriu um algoritmo para resolver o Problema de Carteiro com Vento num grafo par. A idéia da Heurística 1 é tornar o grafo conexo e par, para depois aplicar o algoritmo de Win.

Passo 1. Árvore Geradora Mínima

Acrescentar alguns links ao grafo $G' = (N, E')$ (que em geral é formado de vários componentes), de modo a torná-lo conexo. Isto pode ser feito calculando, e acrescentando a G' uma árvore geradora mínima. Seja G_C tal grafo conexo.

Passo 2. Emparelhamento de Custo Mínimo

Identificar os nós de grau impar no G_C. Calcular um emparelhamento de custo mínimo entre estes nós, considerando as distâncias mínimas entre eles, calculadas no grafo original *G*. Acrescentar links ao grafo, conforme a solução de emparelhamento. Seja G_P o grafo par, resultado desta operação.

Passo 3. Algoritmo de Win para o Grafo Par

Aplicar o algoritmo de Win [Win89] ao grafo G_P, obtendo o grafo orientado G_O. Este grafo é conexo, simétrico, e contém todas as arestas requeridas de *G*, em forma de arcos orientados. Portanto G_O é uma solução de PCRV.

A Heurística 2 é, basicamente, composta das mesmas fases da Heurística 1, porém numa ordem diferente, enquanto que a Heurística 3 é elaborada com duas fases: cálculo de uma Árvore Geradora Mínima, e a solução de um Problema de Transporte. Os autores introduzem também um processo de melhoramento que pode ser aplicado a todas as heurísticas.

Os grafos de teste utilizados por Benavent et al [Ben03.1], para medir o desempenho do algoritmo exato, foram usados também para testar as heurísticas 1, 2 e 3. Para o conjunto de grafos aleatoriamente gerados, os desvios médios em relação à solução ótima foram 5,42%, 8,15% e 7,45%, respectivamente, isso com o emprego da rotina de melhoramento. Os resultados mostram um desempenho relativamente superior da Heurística 1.

Por serem métodos rápidos, os autores recomendam a aplicação dos três métodos, e a escolha do melhor resultado. Neste caso, o desvio médio é reduzido para 3,2%, para o mesmo conjunto de teste.

Algoritmo *Scatter Search* de Benavent et al. [Ben03.2]

No seu Relatório Técnico, Benavent et al [Ben03.2] relatam algumas novas heurísticas inspiradas nas heurísticas de Benavent et al [Ben03.1], e um novo algoritmo de *Scatter Search* (Busca Disseminada) para o PCRV.

As duas heurísticas relatadas são versões modificadas das heurísticas de Benavent et al [Ben03.1]. Os testes computacionais mostraram uma pequena melhoria em relação aos algoritmos originais.

Além destas heurísticas, são apresentados quatro procedimentos Multi-Start, obtidos com a randomização de certas etapas das duas heurísticas. Os algoritmos Multi-Start consistem basicamente na execução iterativa de um método de solução, modificando, cada vez, algum parâmetro do problema. Exemplificando, a árvore geradora utilizada nos procedimentos heurísticos para garantir a conectividade do grafo modificado (Benavent et al. [Ben03.1]), não precisa necessariamente ser mínima. Ela pode ser aleatoriamente construída a partir de um conjunto de arestas candidatas. Os testes computacionais mostraram que alguns dos procedimentos Multi-Start, quando executadas 250 iterações,

produziram soluções com desvio abaixo de 1% da solução ótima, usando os mesmos grafos de teste de Benavent et al [Ben03.1]. O tempo de processamento para os problemas de maior porte foi em geral abaixo de 40 segundos, num computador Pentium IV de 1,7 GHz.

Além das heurísticas acima, os autores apresentam um novo algoritmo baseado na Busca Disseminada. A Busca Disseminada é uma meta-heurística aplicada sobre uma população que tem mostrado resultados satisfatórios na solução de problemas combinatórios difíceis [Glo00].

O processo começa com a construção de um grande conjunto inicial de soluções diversificadas, utilizando uma das heurísticas acima mencionadas. O conjunto é ordenado de acordo com a qualidade das soluções individuais. Daí começa uma pesquisa para obter novas soluções, a partir de combinação de pares de soluções do conjunto. Uma vez gerada uma solução melhor que alguma existentes, ela entra no conjunto, substituindo a pior delas. O processo encerra, quando o conjunto permanecer inalterado depois de combinação de todas as suas soluções.

Os autores relatam testes computacionais para este método, utilizando o mesmo conjunto de grafos de teste de Benavent et al. [Ben03.1], bem como alguns grafos aleatórios de porte maior. Para os grafos maiores, as dimensões médias foram: $|N| = 848$, $|E| = 2522$, e $|E'| = 1149$. Para estes, o desvio médio em relação à solução ótima ficou abaixo de 1,8%, e o tempo médio de processamento em torno de 1500 segundos, num Pentium IV de 1,7 GHz.

4.6 Considerações Sobre as Abordagens Existentes para os Problemas de Carteiro Rural

Os métodos de solução para o Problema de Carteiro Rural, incluindo suas versões, Orientada, Mista, e com Vento, tiveram uma evolução significativa nos anos recentes. As abordagens de solução podem ser classificadas em três grupos, e resumidamente discutidas da seguinte forma:

- Abordagens baseadas na descrição parcial do poliedro de soluções, com introdução de novas classes de desigualdades que ajudam a solução do problema por meio de uma relaxação linear. Nesta linha podem ser destacados os trabalhos de Benavent

et al. [Ben03.1], e Corberán et al. [Cor98], sendo o primeiro com sucesso significativo na obtenção de soluções exatas para problemas de médio porte, conforme indicam os seus resultados computacionais.

- Soluções heurísticas que usam a lógica de acréscimo de links, com objetivo de tornar o subgrafo requerido, par, simétrico, e conexo. Em geral estas abordagens usam ferramentas de otimização, como: Emparelhamento de Custo Mínimo, Fluxo em Redes, e Árvore Geradora Mínima, seguidas por algum método de melhoramento da solução final. Podem ser mencionados os trabalhos de Christofides et al. [Cri86], Eiselt et al. [Eis95.2], Benavent et al. [Ben03.1], Benavent et al. [Ben03.2], e Corberán et al. [Cor00], sendo os últimos dois com destaque na utilização de meta-heurísticas para a fase de melhoramento da solução. Os resultados computacionais mais significativas são os relatados por Benavent et al. [Ben03.2] que obtiveram soluções próximas à ótima para problemas de porte relativamente grande.

- Métodos de transformação, baseados na transformação do Problema de Roteamento de Arcos, em Problema de Roteamento de Nós, com destaque para os trabalhos de Laporte [Lap97], e Corberán et al. [Cor02]. Ambos os trabalhos relatam ineficiência da abordagem na solução de problemas que envolvem um maior número de arestas.

Vale ressaltar que a abordagem adotada nesta Tese para resolver os Problemas de Roteamento de Arcos, a ser discutida nos próximos capítulos, é baseada exatamente num método de transformação. Os testes computacionais relatados nos capítulos 5 e 6, entretanto, confirmam a eficiência do método em todos os casos, inclusive para grafos com grande número de arestas.

V – Contribuições ao Problema de Carteiro Chinês Misto

5.1 Considerações Iniciais

Nesse capítulo será apresentado um método de solução para o Problema de Carteiro Chinês Misto – PCCM. No capítulo 6 este mesmo método será generalizado para o PGR, que, como foi visto, é a instância mais genérica entre todas as formulações não-capacitadas de problemas de roteamento. O método consiste em transformar o Problema de Roteamento de Arcos (PRA) em um Problema de Roteamento de Nós (PRN). O problema resultante é resolvido como um PCV convencional.

Como foram vistos nas Seções 4.4 e 4.6, alguns autores têm sugerido a transformação de PRA's em PRN's. Entretanto, os resultados computacionais, por eles relatados, indicaram um sucesso limitado.

Nessa linha, deve-se mencionar também o trabalho de Pearn et al. [Pea87], em que se descreve um procedimento que transforma o PRA não-orientado, num PRN equivalente. O procedimento proposto associa três novos nós a cada aresta, transformando o grafo $G = (N, E)$ em um grafo com $|N| + 3|E|$ nós. Os autores não apresentam nenhum resultado computacional, fruto desta transformação.

A transformação sugerida por Laporte [Lap97], com o mesmo objetivo, utiliza um procedimento composto de três etapas. Na primeira etapa, cada aresta é substituída por um par de arcos, resultando num grafo completamente orientado; na segunda etapa, é construído um grafo completo em que os nós representam os arcos do grafo da etapa anterior, transformando assim o PRA em um Problema de Caixeiro Viajante Generalizado (PCVG); e finalmente, na terceira etapa, o PCVG é transformado num PCV. O grafo final terá $|A| + 2|E|$ nós. Os testes computacionais relatados indicaram sucesso apenas nos casos de grafos quase orientados (grafos mistos contendo pouquíssimas arestas).

Corberán et al [Cor02] sugerem uma abordagem em que o PRA é transformado diretamente para um Problema de Caixeiro Viajante Assimétrico (PCVA). O método é descrito para o PCRM.

Outro trabalho do gênero é o de Dror e Langevin [Dro97] que transformam o Problema de Carteiro Rural Agrupado (PCRA) em um PCVG. O método é descrito para o caso de grafo completamente orientado. Os testes computacionais relatados estão limitados a problemas de pequeno porte.

O método apresentado nesse capítulo consiste numa transformação do grafo original em um grafo de mesmas dimensões da proposta por Laporte [Lap97]. O procedimento é intuitivo e permite a solução do problema de roteamento como um de PCV padrão no grafo transformado. A abordagem é motivada pelos recentes desenvolvimentos no campo de soluções heurísticas e meta-heurísticas para o PCV. Os resultados computacionais confirmaram a eficiência da abordagem; PCCM's de porte relativamente grande foram resolvidos e soluções próximas às respectivas ótimas foram obtidas em tempos aceitáveis.

Embora o método é destinado a resolver o PGR, para efeitos didáticos, e sem perda de generalidades, na seção seguinte apresenta-se a transformação proposta para o caso particular de PCCM. No capítulo seguinte, o método será apresentado na sua forma geral, aplicável ao PGR.

5.2 Transformação

Nessa seção, será apresentada uma descrição detalhada da transformação proposta, inicialmente para resolver o PCCM. Seja $G = (N, A, E)$ um grafo misto fortemente conexo, como definido na seção 2.2. Como nesse momento o problema em foco é o PCCM, então todos os arcos em A e todas as arestas em E são requeridos. Portanto, para evitar notações complexas, empregam-se A e E também como os conjuntos requeridos.

Relembra-se que $n = |N|$, $r = |A|$ e $m = |E|$ são as cardinalidades dos conjuntos acima, e D a matriz de custos associada a G. O procedimento descrito abaixo transforma o grafo G no grafo G_4, com $r + 2m$ nós.

Passo 1

Construir o grafo $G_1 = (N_1, A_1 \cup E_1)$ a partir do grafo G, como segue:

Inicialmente, sejam $N_1=\varnothing$, $A_1=\varnothing$, e $E_1=\varnothing$.

Para cada link $\ell = (i, j) \in A \cup E$, criar os seguintes componentes em G_1:

(a) dois nós $n_{i\ell}$, $n_{j\ell}$ em N_1;

(b) um arco $(n_{i\ell}, n_{j\ell})$ em A_1, se ℓ é um link do tipo arco; ou

(c) dois arcos $(n_{i\ell}, n_{j\ell})$, $(n_{j\ell}, n_{i\ell})$ em E_1, se ℓ é um link do tipo aresta.

Em qualquer caso, o custo associado a qualquer arco $(n_{i\ell}, n_{j\ell}) \in A_1 \cup E_1$ será o mesmo custo do link original ℓ.

Agora G_1 é um grafo orientado com $r + m$ componentes desconectados, no qual cada nó $i \in N$ tem k copias em N_1, onde k é o número de links conectados ao nó i em G. D_1 é a matriz de custo associado a G_1.

Passo 2

Construir o grafo $G_2 = (N_1, A_1 \cup E_1 \cup B_1)$ a partir de G_1, como descrito abaixo. Seja $P_i = \{i_1, i_2, ..., i_k\}$ o conjunto de cópias do nó $i \in N$ em N_1, e seja inicialmente $B_1 = \varnothing$. Para cada nó $i \in N$ pesquisar o conjunto P_i como segue: para cada par de nós $i_s, i_t \in P_i$, criar o arco (i_s, i_t) em B_1, se, e somente se, i_s é um nó final de qualquer arco em $A_1 \cup E_1$ e i_t é um nó inicial de qualquer arco em $A_1 \cup E_1$. Agora G_2 é um grafo orientado fortemente conexo.

Denominam-se os arcos em B_1 como *arcos de conexão*, os quais podem ser interpretados como passagens nos nós do grafo original, de um link para outro. No capítulo 6 será visto que estes arcos são de grande valia para implementar as restrições de ordem de percurso e/ou imposição de penalidades para conversões indesejáveis no roteiro a ser gerado, com aplicação de custos apropriados aos mesmos. No momento atribui-se custo zero aos arcos de B_1. Seja D_2 a atual matriz de custo associada a G_2.

Passo 3

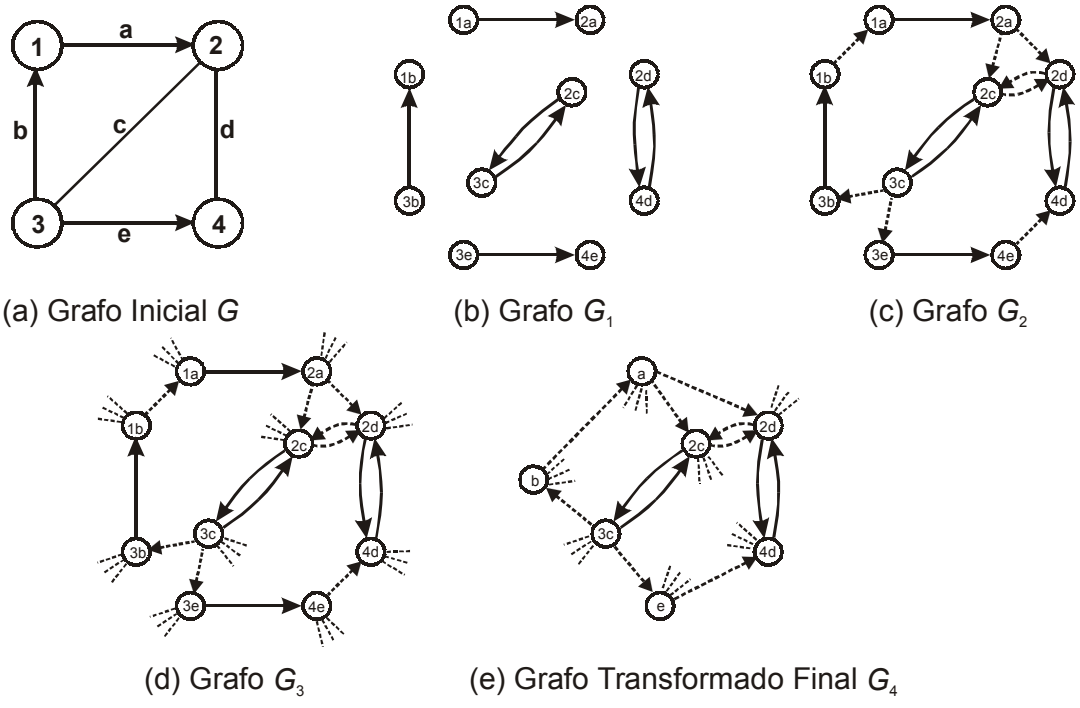
Formar o grafo completo $G_3 = (N_1, A_1 \cup E_1 \cup S_1)$, onde S_1 representa o conjunto de caminhos mínimos entre todos pares de nós $i, j \in N_1$, em G_2, tais que o arco $(i, j) \notin A_1 \cup E_1$. Note que todos os arcos de conexão em B_1 estão agora incluídos em S_1. Entretanto, seus custos podem ter sido reduzidos por meio das distâncias mais curtas. Considere D_3 como sendo a matriz de custos associada a G_3.

Passo 4

Obter o grafo final $G_4 = (N_2, E_1 \cup S_2)$, transformado a partir de G_3, seguindo o procedimento descrito abaixo:

Eliminar todos os arcos em G_3 que pertencem a A_1, unificando seus nós inicial e final, conforme procedimento a seguir. Para cada arco $(i, j) \in A_1$, excluir o nó j, excluir todos os arcos $(v, j) \in S_1$, excluir todos os arcos $(i, v) \in S_1$, e excluir o próprio arco (i, j). Todos os arcos $(j, v) \in S_1$ passam a ser (i, v), com nenhuma alteração em seus custos. Estas modificações reduzem o conjunto de nós N_1 para N_2, e o conjunto de caminhos mínimos S_1 para S_2. Note que cada arco original em *G* é representado agora por um único nó em G_4. Para completar a transformação, alterar o custo de cada arco em E_1 para $-M$, onde M é um número positivo suficientemente grande.

A figura 5.1 mostra um exemplo ilustrativo, no qual a transformação é aplicada sobre o grafo misto *G* (figura 5.1-a), visando resolver o PCCM. As figuras 5.1-b, 5.1-c e 5.1-d mostram os grafos transformados ao final de cada passo, e 5.1(e) mostra o grafo transformado final.



(a) Grafo Inicial *G* (b) Grafo G_1 (c) Grafo G_2

(d) Grafo G_3 (e) Grafo Transformado Final G_4

Figura 5.1. Exemplo para a transformação.

As operações efetuadas nessa transformação são todas simples, exceto o cálculo de caminhos mínimos entre todos os pares de nós, efetuado no passo 3, cuja complexidade é da ordem $O(v^3)$, onde $v = 2|A| + 2|E|$ é o número de nós do grafo transformado nos passos iniciais do algoritmo. Logo, este mesmo valor pode ser considerado como a complexidade global do algoritmo de transformação.

5.3 A Solução do PCV

Uma vez efetuada a transformação, a etapa seguinte consiste em resolver o problema do caixeiro viajante no grafo transformado. Inicialmente mostra-se a correspondência entre o circuito de carteiro em G e um circuito hamiltoniano em G_4.

Como foi visto, o grafo $G = (N, A, E)$ é transformado no $G_4 = (N_2, E_1 \cup S_2)$. Cada aresta em G é representada por um par de arcos contrariamente orientados em G_4. E_1 é o conjunto de tais pares. Ainda mais, cada arco em G é representado por um único nó em G_4. Seja $N_a \subseteq N_2$ o conjunto de tais nós. Seja $N_e \subseteq N_2$ os demais nós em G_4 que formam os nós terminais de E_1. Obviamente, $N_a \cap N_e = \varnothing$ e $N_a \cup N_e = N_2$.

Considera-se agora C_C como um Circuito de Carteiro em G, o qual passa por todos os arcos e arestas, e possivelmente com algumas cópias extras dos links. Ao traçar-se este circuito no grafo transformado G_4, obter-se-á um circuito C_H, o qual contém pelo menos um arco de cada par em E_1, e todos os nós em N_a. Como o circuito conterá um arco de cada par em E_1, então conterá todos os nós em N_e. Logo o circuito conterá todos os nós em $N_a \cup N_e = N_2$. Isto significa que C_H é um Circuito Hamiltoniano em G_4. Vale ressaltar que as cópias extras dos links num circuito de carteiro ótimo em G formam caminhos mínimos entre alguns pares de nós. Em G_4, estes caminhos são representados apenas por passagens através dos arcos em S_2, sem passagem adicional por nenhum nó em N_2.

Pode-se verificar que a recíproca também é verdadeira: isto é, um Circuito Hamiltoniano em G_4 corresponde a um Circuito de Carteiro em G. É fácil verificar que

qualquer solução ótima, ou próxima a ótima do PCV em G_4 provê um circuito que satisfaz as seguintes características: contém um arco de cada par em E_1, e todos os nós em N_a. De fato, como G_4 é um grafo completo, nele existe um Circuito Hamiltoniano que contém todos os nós em N_2 exatamente uma vez (incluindo N_a). E se tal circuito é uma solução ótima, ou próxima a ótima de PCV, ele contém um arco de cada par de arcos em E_1, justamente porque estes se apresentam com custo negativo elevado. Isto significa que o traçado do circuito correspondente em G, irá conter todas as arestas em E e todos os arcos em A. Por conseguinte, a solução de PCV em G_4 equivale a solução de PCCM em G.

Sejam $H \subset E_1 \cup S_2$ o conjunto de arcos coberto por um circuito hamiltoniano mínimo em G_4, e Z_H o custo total associado ao circuito. Então H deve conter m arcos pertencentes a E_1, onde $m = |E|$, e alguns outros arcos pertencentes a S_2, os quais correspondem às coberturas extras de links no Circuito de Carteiro, ou simplesmente às passagens nos nós (arcos de conexão). Com objetivo de neutralizar os custos negativos incluídos em Z_H, calcula-se $Z_G = Z_H + m \cdot M$, o qual representa o custo do circuito hamiltoniano, expurgado o custo dos arcos em E_1. Portanto, Z_G representa o custo total das passagens extras num Circuito de Carteiro em G.

Uma discussão acerca das técnicas de solução para o PCV está fora do escopo deste trabalho. Em princípio, qualquer método de solução para o caso assimétrico desse problema pode ser adotado. Podem ser métodos exatos ou aproximados. Entretanto, se tratando de métodos aproximados, a solução gerada para o PCV deve ser próxima à ótima, não apenas para garantir boa qualidade de solução para o PCCM, mas também para garantir a viabilidade da mesma. Como foi visto, o Circuito Hamiltoniano em G_4 deve conter um arco de cada par de arcos que corresponde a uma aresta (conjunto E_1). O atrativo para isso é o custo negativo elevado para tais arcos. Uma solução longe da ótima pode não contemplar esse requisito fundamental.

De fato o método de transformação proposto nesse trabalho é motivado pelos recentes melhoramentos no campo de soluções para o PCV. Johnson et al. [Joh97] e Voudouris et al. [Vou99] têm sugerido soluções aproximadas que foram aplicadas com sucesso aos PCV's de grande porte. A meta-heurística Busca Local Dirigida, desenvolvida pelos

últimos autores, tem encontrado soluções próximas à ótima para problemas de médio e grande porte, em tempos relativamente curtos.

Para os testes computacionais foi adotada uma implementação da busca local dirigida elaborada por Rodrigues [Rod00]. Este autor nos seus testes resolve PCV's de até 14.000 nós, com soluções ótimas conhecidas. O relato mostra a obtenção de soluções próximas à ótima em todos os casos testados.

5.4 Testes Computacionais para o PCCM

O procedimento descrito acima foi implementado num Computador Pessoal equipado com processador Pentium IV – 2.0 GHz. Não foi possível testar o método com grafos utilizados por outros autores, visando a comparação direta dos resultados computacionais com trabalhos já publicados. A razão disso foi a indisponibilidade de tais grafos de testes pelos respectivos autores.

Os testes foram feitos em grafos pseudo-manhattan, gerados aleatoriamente. Estes foram construídos numa grade de $n = p \times q$ nós, em que cada nó pode estar em contato com um máximo de oito nós, exceto para os nós de fronteira e canto, para os quais este número é reduzido para cinco e três nós, respectivamente. Um número predefinido de arcos e arestas conecta aleatoriamente os nós, assegurando a condição de conectividade para o grafo. Como o grafo é do tipo pseudo-manhattan, então links que conectem pares de nós na posição diagonal são permitidos. A figura 5.2 mostra um exemplo típico de um grafo pseudo-manhattan, com 12 nós, numa grade de 3×4.

Os testes foram realizados em cinco grupos, com grafos de 100, 200, 300, 400 e 500 nós cada um. Para cada grupo, cinco grafos foram gerados, com percentuais de 0%, 25%, 50%, 75% e 100% de links orientados (arcos). Portanto, os testes contemplam não apenas os casos mistos, mas também os dois casos extremos – grafos totalmente orientados e totalmente não-orientados. Não foram consideradas as restrições de conversão nos nós.

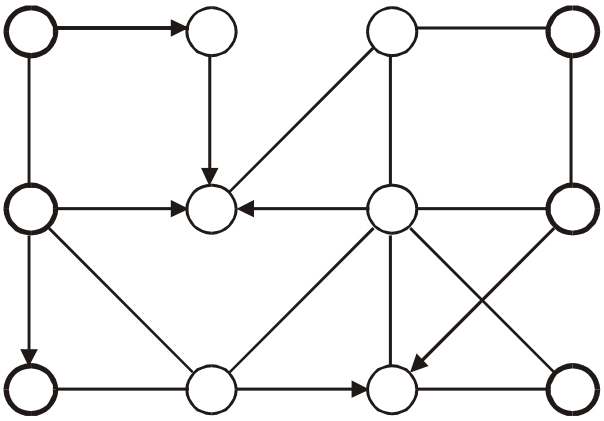


Figura 5.2 – Exemplo de um Grafo Misto Pseudo-Manhattan

Estes grafos, depois de gerados passaram pelo algoritmo de transformação, descrito na seção 5.2, e foram submetidos a uma rotina de PCV, baseado no método de Busca Local Dirigida [Rod00].

A Tabela 5.1 mostra informações detalhadas sobre os testes computacionais. Para cada instância dos testes foram providenciados dois limites: um limite inferior e um superior para a solução ótima. O limite inferior para a distância do circuito de carteiro foi calculado, resolvendo o PCC no grafo não-orientado, derivado do grafo original, ignorando todas as orientações dos arcos. E o limite superior para a solução ótima foi obtido de uma rotina especial inclusa no gerador de grafos aleatórios, a qual permite construir um grafo juntamente com uma boa solução viável para o PCCM. A experiência mostrou que esta solução fica muito próxima à ótima e provê um bom parâmetro de comparação.

Como pode ser visto na Tabela 5.1, as soluções obtidas pelo algoritmo proposto apresentam um desvio máximo de 7,9%, se comparadas com os correspondentes limites inferiores, dentre os casos mistos testados. Vale ressaltar que para os problemas totalmente não-orientados, o limite inferior coincide com a solução ótima; entretanto, para os casos totalmente orientados, o limite inferior pode se situar muito abaixo da respectiva solução ótima e deixa de ser um parâmetro eficiente de comparação. Comparando com as boas soluções conhecidas, quase em todos os casos elas são igualadas, ou superadas pelo método proposto. Apenas em três casos tais marcas não foram atingidas, com o desvio máximo registrado em 0,9%.

Tabela 5.1. Testes Computacionais nos Grafos Pseudo-Manhattan

Nome do Grafo	Núm. Nós (1)	Núm. Links (2)	% P_a (3)	Limite Inferior (4)	Limite Superior (5)	Solução Obtida (6)	t_1 Seg. (7)	t_2 Seg. (8)	t_3 Seg. (9)	% Desv. (10)
Mix100A00	100	200	0	21410	21410	21410	0.27	2.4	3	0.0
Mix100A25	100	200	25	21440	21537	21537	0.36	3.0	3	0.5
Mix100A50	100	200	50	21673	22138	22103	0.24	3.9	4	2.0
Mix100A75	100	200	75	21444	22203	22174	0.21	0.7	1	3.4
Mix100A100	100	200	100	21187	24205	24205	0.12	0.1	0	14.2
Mix200A00	200	400	0	42976	42976	42976	1.68	23.	25	0.0
Mix200A25	200	399	25	43687	44144	43945	2.16	80.	82	0.6
Mix200A50	200	400	50	42778	44405	44261	1.44	38.	39	3.5
Mix200A75	200	400	75	42717	46191	46097	0.96	14.	15	7.9
Mix200A100	200	400	100	42451	47193	47193	0.72	0.6	1	11.2
Mix300A00	300	600	0	64612	64612	64612	6.0	58.	64	0.0
Mix300A25	300	599	25	64261	65006	64734	7.20	173.	180	0.7
Mix300A50	300	600	50	64831	66517	66502	5.76	154.	160	2.6
Mix300A75	300	600	75	64240	68190	68080	3.36	23.	26	6.0
Mix300A100	300	600	100	63504	72673	72673	2.40	3.	5	14.4
Mix400A00	400	800	0	86208	86208	86208	14.2	227.	241	0.0
Mix400A25	400	798	25	85865	86214	86394	17.3	177.	194	0.6
Mix400A50	400	800	50	86815	89079	89014	14.4	207.	221	2.5
Mix400A75	400	800	75	86233	91592	91494	6.7	109.	116	6.1
Mix400A100	400	799	100	85189	96517	96326	5.8	6.	12	13.1
Mix500A00	500	999	0	107470	107470	108435	27.4	305.	332	0.9
Mix500A25	500	1000	25	107803	108370	108649	32.2	335.	367	0.8
Mix500A50	500	1000	50	107770	110640	110598	21.8	71.	93	2.6
Mix500A75	500	1000	75	107707	114268	114268	14.2	91.	105	6.1
Mix500A100	500	1000	100	106191	122080	122080	12.0	7.	19	15.0

(1) Número de nós;
(2) Número de links (arcos+arestas);
(3) Percentual de links orientados (arcos);
(4) Limite inferior no custo do circuito de PCCM;
(5) Limite superior (Uma boa solução conhecida);
(6) Valor da solução, obtido pelo método proposto;
(7) Tempo de CPU, usado no procedimento de transformação;
(8) Tempo de CPU, usado no procedimento de PCV;
(9) Tempo de CPU total;
(10) Percentagem de desvio entre a solução obtida e o limite inferior.

Como o método de Busca Local Dirigida, utilizado para resolver o PCV é uma técnica iterativa, isso permite a obtenção de soluções menos aproximadas em tempos mais curtos. Por exemplo, resolvendo o grafo misto Mix500A25 (com 500 nós, 250 arcos e 750 arestas), ao cabo de t_2= 60 segundos de processamento, foi registrada uma solução viável que se situa a 7,2% do limite inferior.

O tempo consumido pelo processo de transformação em geral é bem menor do que o tempo para o procedimento do PCV. Para os grafos de teste com 500 nós e 1000 links, o tempo médio para concluir o processo de transformação foi de 21,5 segundos. Enquanto o tempo médio usado pela rotina de PCV para este mesmo grupo de grafos foi de 161,8 segundos.

Outra análise que pode ser feita é em relação à evolução do tempo de processamento, em função de percentual de links orientados. A Tabela 5.2 mostra o tempo médio de processamento da rotina do PCV, não apenas em função do percentual de arcos, mas também, em função de número total de links que compõe os grafos. Como se vê nessa Tabela, o método é extremamente rápido para grafos completamente orientados, com tempo médio de 3,3 segundos para todos os casos dessa categoria. Entretanto, para as instâncias com maior percentual de arestas, o tempo aumenta significativamente. Uma parte desta diferença pode ser justificada com o fato de que cada arco se representa por um só nó no grafo transformado, enquanto cada aresta contribui com dois nós neste. Outra parte da diferença é devida a própria sensibilidade do método usado para a solução do PCV, cujo estudo não faz parte dos objetivos desta pesquisa.

Tabela 5.2. Tempo de processamento da rotina do PCV, em função do número de links, e percentual de arcos

% arcos - *P*	Número de links					Tempo Médio
	200	400	600	800	1000	
0	2,4	23	58	227	305	123,1
25	3	80	173	177	335	153,6
50	3,9	38	154	207	71	94,8
75	0,7	14	23	109	91	47,5
100	0,1	0,6	3	6	7	3,3
Tempo Médio	2,0	31,1	82,2	145,2	161,8	84,5

O método pode ser aplicado eficientemente aos grafos genéricos. Entretanto, pode-se verificar que ambos os tempos de processamento, t_1 e t_2, são fortemente afetados pela dimensão da matriz de custo transformada. Como foi visto, esta dimensão é igual a $r+2m$, onde $r = |A|$ e $m = |E|$, razão pela qual o método não é indicado para grafos densos.

O problema de maior porte testado foi de um grafo pseudo-manhattan, com 1000 nós e 2000 links, dos quais 50% eram orientados. A solução obtida para este ficou 9,7% acima do limite inferior e 5,2% acima da melhor solução conhecida, no tempo total de t_3=473 segundos.

É possível fazer uma comparação indireta do método proposto com os algoritmos Mixed I e Mixed II de Pearn e Chou [Pea99]. Eles realizaram testes computacionais para estes métodos, utilizando um conjunto de 120 grafos mistos gerados aleatoriamente, todos na faixa entre 10 a 35 nós, incluindo 40 problemas com $70\% < P \leq 100\%$, 40 problemas com $40\% < P \leq 70\%$ e 40 problemas com $0\% < P \leq 40\%$, onde P representa o percentual de links orientados (arcos), em relação ao número total de links $\left(P = \left(r * 100\%\right) / \left(r + m\right)\right)$. Para efeito de comparação, foi gerado um conjunto de 27 grafos aleatórios, com dimensões e características semelhantes às acima apresentadas. Os testes com estes grafos estão relatados na Tabela 5.3.

A Tabela 5.4 compara o desempenho dos algoritmos Mixed Aperfeiçoados e o Método proposto. Vale ressaltar que os algoritmos Mixed I Aperfeiçoado e Mixed II Aperfeiçoado são os de melhor desempenho relatado entre todas as versões dos algoritmos Mixed, tendo em vista que a versão mais recente, a de ½-aproximação, devido a Raghavachari e Veerasamy [Rag99], não vem acompanhada de testes computacionais. A comparação mostra que, pelo menos para as amostras utilizadas em cada caso, as abordagens são quase equivalentes quando se trata de grafos mistos que se aproximam a não-orientados $\left(0\% < P \leq 40\%\right)$. Entretanto, para os grafos que contém um maior percentual de arcos ($40\% < P \leq 70\%$ e $70\% < P \leq 100\%$) o método proposto se apresenta superior. Essa diferença se destaca mais nos piores casos de cada abordagem. Nessa matéria, a presente abordagem mostrou-se ter um comportamento regular, sem nenhum resultado ruim surpreendente. Essa comparação não é conclusiva, uma vez que os testes foram feitos sobre diferentes conjuntos de grafos; logo, as diferenças podem ser também explicadas pelas estruturas dos grafos em cada conjunto de teste.

Tabela5.3. Testes computacionais para efeito de comparação

Nome do Grafo	Núm. Nós (1)	Núm. Links (2)	% P_a (3)	Limite Inferior (4)	Limite Superior (5)	Solução Obtida (6)	t_3 Seg. (9)	% Desv. (10)
A10P95	10	21	0,95	2248	2583	2583	0,003	14,9
A10P85	10	21	0,85	2309	2504	2504	0,006	8,4
A10P75	10	21	0,75	2299	2504	2504	0,006	8,9
A20P95	20	54	0,95	5750	5971	5971	0,015	3,8
A20P85	20	54	0,85	5670	5891	5891	0,033	3,9
A20P75	20	54	0,75	5771	5917	5917	0,030	2,5
A35P95	35	105	0,95	10932	11721	11721	0,060	7,2
A35P85	35	106	0,85	11115	11519	11519	0,090	3,6
A35P75	35	106	0,75	11075	11382	11382	0,114	2,8
A10P65	10	21	0,65	2384	2430	2430	0,006	1,9
A10P55	10	21	0,55	2305	2305	2305	0,006	0,0
A10P45	10	21	0,45	2397	2397	2397	0,015	0,0
A20P65	20	55	0,65	5833	6025	6025	0,045	3,3
A20P55	20	55	0,55	5760	5823	5808	0,159	0,8
A20P45	20	54	0,45	5671	5704	5704	0,129	0,6
A35P65	35	106	0,65	11138	11508	11508	0,153	3,3
A35P55	35	106	0,55	11139	11324	11324	0,477	1,7
A35P45	35	106	0,45	11066	11166	11166	0,411	0,9
A10P35	10	21	0,35	2331	2372	2331	0,009	0,0
A10P25	10	21	0,25	2385	2421	2385	0,009	0,0
A10P15	10	21	0,15	2304	2304	2304	0,010	0,0
A20P35	20	54	0,35	5843	6151	6151	0,141	5,3
A20P25	20	55	0,25	5814	5958	5958	0,078	2,5
A20P15	20	55	0,15	5882	5882	5882	0,126	0,0
A35P35	35	106	0,35	11002	11045	11045	0,537	0,4
A35P25	35	106	0,25	11171	11233	11171	0,591	0,0
A35P15	35	106	0,15	11053	11095	11053	0,465	0,0

Tabela 5.4. Comparação entre os algoritmos Mixed e o Método Proposto

Percentual de Arcos *P*	Valores Obtidos Acima do Limite Inferior	Mixed I (*) Aperfeiçoado	Mixed II (*) Aperfeiçoado	Algoritmo Proposto
$70\% < P \leq 100\%$	% Desvio Médio	29,42	25,90	6,2
	% Desvio no Pior Caso	83,33	62,67	14,9
$40\% < P \leq 70\%$	% Desvio Médio	6,32	4,95	1,4
	% Desvio no Pior Caso	42,73	42,73	3,3
$0\% < P \leq 40\%$	% Desvio Médio	0,75	0,13	0,9
	% Desvio no Pior Caso	16,04	3,21	5,3

(*) Extraídos dos testes computacionais realizados por Pearn e Chou [Pea99].

Quanto à eficiência comparativa dos métodos e seu desempenho na solução de problemas de maior porte, não se pode fazer comentários. Infelizmente, os testes relatados para os algoritmos Mixed se limitam apenas a problemas de pequeno porte. Portanto, não há idéia sobre o tempo de processamento destes na medida em que aumenta o tamanho do problema.

Pelos resultados dos testes computacionais vistos acima, pode-se concluir que o algoritmo proposto preenche os parâmetros desejáveis de um bom método para ser usado na prática, na solução de PCCM, considerando:

- qualidade de soluções obtidas, em geral ótimas, ou muito próximas a elas;
- capacidade de resolver problemas de porte relativamente grande; e
- comportamento regular em todos os casos testados, sem surpresas de pior caso.

VI – Contribuições ao Problema Geral de Roteamento

6.1 Considerações Iniciais

O método apresentado no Capítulo anterior será estendido para o Problema Geral do Roteamento – PGR. Esta instância é a mais genérica entre todas as formulações não-capacitadas de problemas de roteamento, a qual tem como casos particulares vários outros problemas anteriormente formulados, entre eles o PCCM e o PCR.

No caso específico do Problema de Carteiro Rural – PCR, como foi visto no Capítulo 4, houve recentes progressos em três linhas de abordagens: métodos exatos, com tentativas baseadas na descrição parcial do poliedro de soluções; métodos heurísticos que usam a lógica de acréscimo de links; e os métodos de transformação, que transformam o PRA em um PRN equivalente. Os melhores resultados computacionais obtidos foram para as primeiras duas abordagens. Os sucessos relatados para os métodos de transformação foram bastante limitados, principalmente devido à ineficiência quando o grafo misto contém relativamente mais arestas que arcos.

Na continuação deste Capítulo, serão abordadas as restrições de conversão nos vértices (uma restrição inerente a qualquer problema de roteamento formulado numa malha urbana), raramente tratadas na literatura, e em seguida generalizando o PCCM para obtenção de uma solução do Problema Geral de Roteamento. Serão relatados os experimentos computacionais, para o caso do Problema de Carteiro Rural Misto com Conversões Penalizadas, e os resultados obtidos são comparados com alguns já publicados por outros autores.

6.2 Restrições nos Vértices

Nesta seção serão consideradas as restrições de ordem de percurso nos nós de um grafo. Quando o grafo representa uma malha urbana, tais restrições podem representar as conversões proibidas nos cruzamentos, como os retornos em U e as conversões à esquerda. Embora elas estejam entre as mais fortes nos problemas reais de roteamento, e possam ser associadas a qualquer formulação particular dos mesmos, são freqüentemente

negligenciadas na formulação desses problemas. Quase todas as abordagens vistas nos capítulos 3 e 4, ignoram tais restrições. Um trabalho de destaque sobre a penalização de conversões indesejáveis é devido a Corberán et al. [Cor02]. A seguir será mostrado como estas restrições podem ser contempladas pelo método proposto.

Em contraste à abordagem adotada nesta Tese, a maioria das técnicas para os problemas de roteamento de arcos, incluindo aquelas para o PCCM e o PCR, resolve o problema em dois estágios distintos: no primeiro é determinado de maneira ótima o grafo aumentado; e no segundo é construída uma rota (um circuito euleriano) que cobre os componentes requeridos do grafo. Quando não existem restrições adicionais, a construção desse circuito, como foi visto, é uma tarefa trivial, e o algoritmo sugerido por Edmonds e Johnson [Edm73] resolve o problema. Entretanto, na presença de restrições de conversão nos vértices, o segundo estágio se torna um problema bastante relevante. O grafo aumentado obtido no primeiro estágio pode requerer um novo aumento, com o objetivo de atender às restrições de percurso estabelecidos nos nós, como pode ser visto no exemplo abaixo.

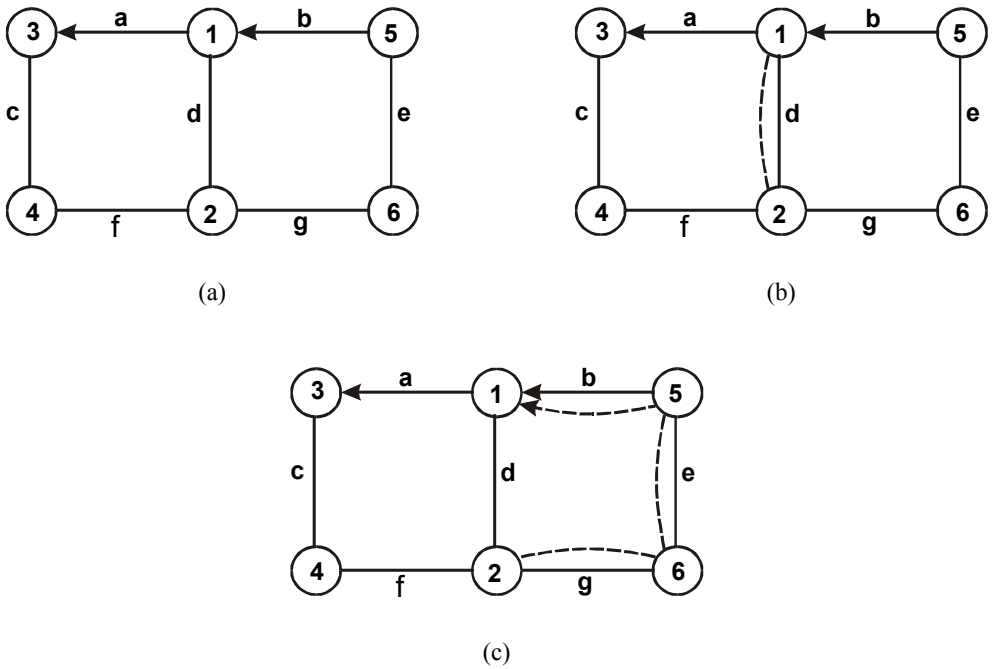


Figura 6.1

Suponha que no grafo misto da figura 6.1(a) os retornos em U nos nós 1 e 2 são proibidos. A figura 6.1(b) representa o grafo aumentado da solução ótima, obtida por

qualquer método, visando a solução para o PCCM. Embora 6.1(b) seja um grafo euleriano, nele não existe nenhum Circuito de Carteiro, iniciando, por exemplo, no nó 1, que satisfaça as restrições adicionais de conversão. Qualquer tentativa resulta num retorno em U. Considere, por exemplo, as seqüências (*a, c, f, g, e, b, d, d'*) e (*d, g, e, b, a, c, f, d'*). Nesta última, o retorno está implícito, uma vez que o circuito começa e termina com a mesma aresta. De fato, qualquer solução viável necessita a duplicação adicional de outras arestas, como por exemplo, aquelas mostradas na figura 6.1(c). Neste último grafo aumentado, um circuito sem retorno U pode ser construído da seguinte forma: (*a, c, f, g, e, b, d, g', e', b'*).

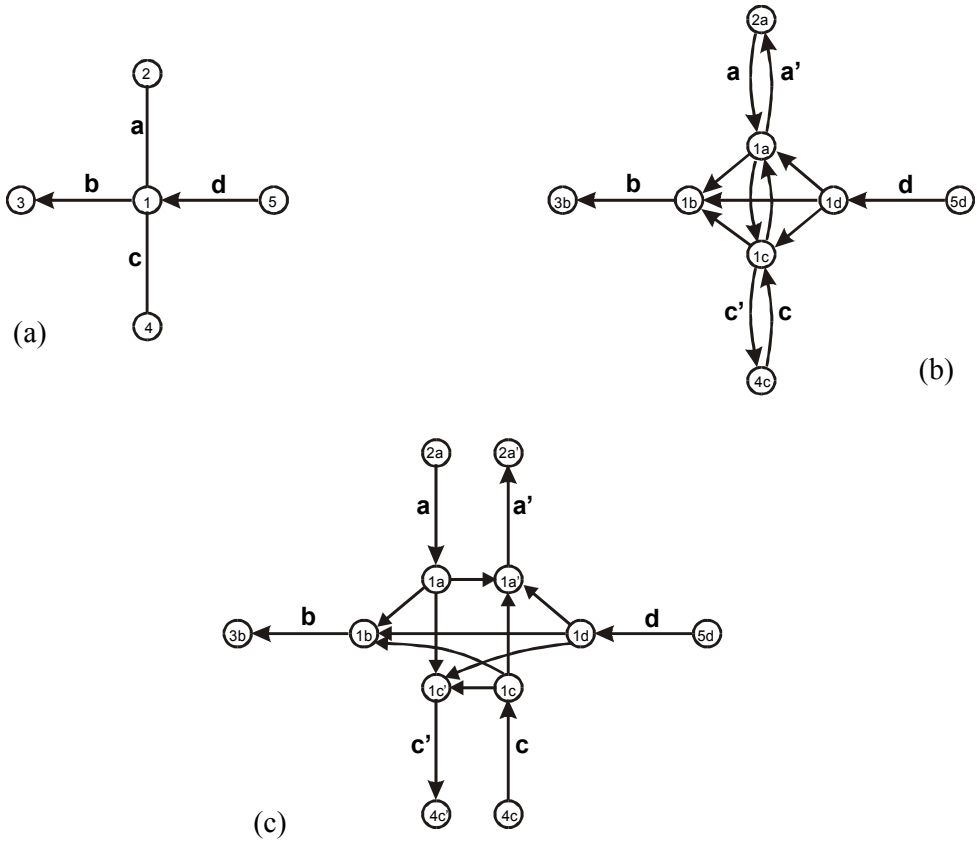
Na abordagem proposta, as restrições nos vértices e a construção do grafo aumentado são tratadas num único estágio. A chave para isso é achar uma forma apropriada de atribuição de custos aos arcos de conexão, introduzidos no processo de transformação (veja sessão 5.2). Como foi visto, os arcos de conexão podem ser interpretados como passagens nos nós do grafo original, de um link para outro. Portanto, se num nó a passagem de um link para outro é indesejável, o arco de conexão associado poderá conter um custo correspondente ao grau da inconveniência.

Desta forma, a matriz de custos, obtida ao final do processo de transformação, pode ser estruturada, não apenas pelos custos dos links, mas também pelas penalidades de passagens nos nós. Por conseguinte, a solução de PCV, no grafo transformado, fornecerá uma solução viável completa para o problema de roteamento de arcos, a qual minimizará o custo total do percurso, relacionado às distâncias e às penalidades de conversão. Isto é uma das principais potencialidades do método proposto. Entretanto, abaixo será mostrado que uma atribuição direta de custos aos arcos de conexão não resolverá o problema.

Considera-se uma situação típica mostrada na Figura 6.2(a). O exemplo pode ser visto como um subgrafo obtido de um grafo maior G. Suponha que nesse grafo os retornos em U e a conversão à esquerda descritos pelas seqüências $(c,1,b)$, $(a,1,a)$ e $(c,1,c)$ devam ser evitados. Aplicando os dois primeiros passos do algoritmo proposto no Capítulo 5, obtém-se o grafo da Figura 6.2(b), que representa o grafo transformado G_2. Nesse grafo, se for suprimido o arco de conexão $(1c,1b)$, ou atribuído um custo elevado ao mesmo, com objetivo de proibir uma conversão à esquerda, isso não irá inibir uma passagem do nó $1c$ para $1b$ numa solução de PCV no grafo G_2. No passo 3 do procedimento de transformação serão calculados os caminhos mínimos, e o caminho $(1c,1a,1b)$ poderá substituir o referido

arco de conexão. Além do mais, o Grafo G_2 (Figura 6.2(b)) não possibilita a penalização para os retornos em U.

Figura 6.2.



A forma viável para inserir penalidades devido às conversões indesejáveis é efetuar uma "expansão" adicional do grafo G_2, como descrita a seguir. Cada aresta do grafo original será substituída por dois arcos, porém estes arcos terão nós terminais distintos, um do outro. A figura 6.2(c) mostra o grafo expandido G_2', em que os arcos de conexão $(1a,1a')$ e $(1c,1c')$ representam os retornos em U, e $(1c,1b)$ a conversão a esquerda. Agora é possível suprimir estes arcos de conexão referentes às conversões proibidas, ou aplicar a eles penalidades apropriadas. A partir disso, os caminhos mais curtos devem ser calculados dos nós $1a$, $1c$ e $1d$ para todos os demais nós. Estes caminhos serão as conversões viáveis, que substituirão cada conversão proibida, e utilizarão possivelmente uma cadeia mais longa de links, do que um simples arco de conexão.

Todavia, o método proposto não tolera esta expansão adicional: Não há como resolver um Problema de Roteamento de Nós nesse grafo, equivalente ao Problema de Roteamento de Arcos. A idéia é, após o calculo de caminhos mínimos no grafo expandido da Figura 6.2(c), contrair o grafo e voltar para a forma da Figura 6.2(b). De fato, os caminhos mínimos calculados no grafo expandido, dos nós $1a$, $1c$ e $1d$ para todos os demais, serão aproveitados no grafo inicialmente transformado.

Os custos de arcos de conexão no grafo expandido, mesmo daqueles que não correspondem às proibições acima, podem ser ajustados convenientemente, de modo a refletirem o custo real de passagem em cada caso.

Vale ressaltar que uma vez calculados os custos no grafo expandido e depois transportados para o grafo transformado inicial, não haveria mais o risco do algoritmo de PCV cometer uma conversão proibida. Por exemplo, na Figura 6.2(b) não haveria como o arco de conexão $(1c,1b)$ ser substituído por uma rota, tal como $(c,1c,1a,1b,b)$, porque numa solução ótima de PCV o arco de conexão $(1c,1a)$ será usado, se, e somente se, o arco a', o qual tem um custo negativo grande, é percorrido na seqüência. Caso contrário, o link a deixaria de ser utilizado em qualquer direção. Caso esta última situação venha ocorrer numa solução ótima (ou próximo à ótima) do PCV, isso é, se com a supressão de arcos de conexão indesejáveis (ou atribuindo custo elevado a eles) alguma aresta (com custo negativo) não participar na solução final, conclui-se, então, que não há nenhuma solução viável de carteiro que atenda às restrições nos vértices. Por outras palavras, isto significa, também, que o grafo deixou de ser fortemente conexo devido às restrições nos vértices.

O algoritmo a ser apresentado na próxima seção, incorpora a abordagem acima descrita ao método de transformação já visto, e ao mesmo tempo generaliza o método para o caso do PGR.

6.3 O Problema Geral de Roteamento com Conversões Penalizadas

Como foi visto no Capítulo 2, o Problema Geral de Roteamento – PGR, é o caso mais genérico entre todas as formulações não-capacitadas dos problemas de roteamento. Consiste em achar um circuito de custo mínimo que cobre um subconjunto de arcos, um

subconjunto de arestas e um subconjunto de nós, de um dado grafo misto. Nessa seção será apresentado um método de transformação do Problema Geral de Roteamento com Conversões Penalizadas - PGRCP, para um PRN. O procedimento é uma generalização do método descrito no Capítulo anterior.

Dado um grafo misto fortemente conexo $G = (N, A, E)$, com N', A' e E' sendo os conjuntos requeridos, conforme definição anterior. Seja $N_T \subseteq N$ o conjunto de nós que aparecem como nó terminal de qualquer link em $A' \cup E'$. Logo, todos os nós em N_T são implicitamente requeridos, e forçosamente serão visitados por qualquer solução de PGR. Seja $N_X \subseteq N'$ o conjunto de todos os nós requeridos que não estão cobertos por qualquer link em $A' \cup E'$. Portanto, tem-se um conjunto maior de nós requeridos $N_R = N_X \cup N_T$.

O procedimento transforma inicialmente o grafo *G*, com *n* nós, em um grafo com $2r + 4m$ nós, onde $n = |N|$, $r = |A|$ e $m = |E|$. Porém, o grafo transformado final será reduzido para $n_x + r' + 2m'$ nós, onde $n_x = |N_X|$, $r' = |A'|$ e $m' = |E'|$. Assim, o procedimento de transformação, visando o PGRCP, pode ser resumido nos seguintes passos:

Passo 1

Construir o grafo $G_1 = (N_1, A_1 \cup E_1)$ a partir do grafo G, conforme descrito a seguir. Sejam inicialmente, $N_1 = \varnothing$, $A_1 = \varnothing$, e $E_1 = \varnothing$.

Para cada arco $a = (i, j) \in A$, criar os seguintes componentes em G_1:

(a) dois nós n_{ia}, n_{ja} em N_1;

(b) um arco (n_{ia}, n_{ja}) em A_1.

Para cada aresta $e = (i, j) \in E$, criar os seguintes componentes em G_1:

(a) quatro nós n_{ieu}, n_{iev}, n_{jeu}, e n_{jev} em N_1;

(b) dois arcos (n_{ieu}, n_{jeu}), (n_{jev}, n_{iev}) em E_1.

O custo associado a qualquer um dos arcos criados em $A_1 \cup E_1$ será o mesmo custo do link original em $A \cup E$.

Observe que o grafo G_1 é constituído, não apenas dos links requeridos, mas também dos não-requeridos. A idéia da transformação do grafo inteiro, nesse passo, é permitir a inserção de restrições nos vértices, incluindo aquelas que incidem aos nós não-requeridos,

que mesmo assim, podem participar da rota final. Nesse estágio, G_1 é um grafo orientado com $r + 2m$ componentes desconectados.

Seja $P_i = \{i_1, i_2, ..., i_k\}$ o conjunto de cópias do nó $i \in N$ em N_1, como criadas acima. denomina-se P_i como a *imagem* do nó $i \in N$ em N_1. Para cada conjunto P_i, $i \in N_X$, escolhe-se um único nó $i_x \in P_i$ para representar o respectivo nó requerido *i* no conjunto N_1. Seja $N_{1X} \subset N_1$ o conjunto dos nós escolhidos desta maneira. Todos os nós em N_{1X} permanecerão no grafo transformado final a fim de participarem do circuito de PCV. A questão de qual nó em cada conjunto P_i deve ser escolhido, nos problemas reais de distribuição pode ser respondida levando em conta a localização exata do ponto a ser servido na rede. Por exemplo, na figura 6.2 (c), embora o nó 1 seja representado por 6 nós cópias, cada um destas terá uma localização distinta da outra no grafo original. Se o nó 1 é requerido, visitá-lo na posição 1*c* pode ser diferente de visitá-lo na posição 1*c*′.

Os nós do conjunto N_1 podem ser classificados em quatro grupos:

$N_{1A} \subseteq N_1$ é o conjunto de nós associados aos arcos requeridos, A';

$N_{1E} \subseteq N_1$ é o conjunto de nós associados às arestas requeridas, E';

$N_{1X} \subseteq N_1$ é o conjunto de nós associados ao conjunto N_X, e

$N_{1D} \subseteq N_1$ é o conjunto de nós que não estão em nenhum dos conjuntos N_{1A}, N_{1E}, e N_{1X}.

O conjunto $N_{1R} = N_{1A} \cup N_{1E} \cup N_{1X}$ forma a imagem dos nós implícita ou explicitamente requeridos em G_1.

Denominar, também, A_{1R} e E_{1R} como os conjuntos de arcos em G_1 associados, respectivamente, aos conjuntos requeridos A' e E'.

Seja D_1 a matriz de custos associada a G_1.

Passo 2

Construir o grafo $G_2 = (N_1, A_1 \cup E_1 \cup B_1)$ a partir de G_1, como descrito a seguir. Seja B_1 o conjunto de arcos de conexão, inicialmente $B_1 = \varnothing$. Para cada nó $i \in N$ pesquisar o conjunto $P_i = \{i_1, i_2, ..., i_k\}$ (imagem do nó *i* em N_1) como segue: Para cada par de nós

$i_s, i_t \in P_i$, criar o arco (i_s, i_t) em B_1, se, e somente se, i_s é um nó final de qualquer arco em $A_1 \cup E_1$ e i_t é um nó inicial de qualquer arco em $A_1 \cup E_1$.

Agora, G_2 é um grafo orientado e fortemente conexo. O conjunto B_1 é formado pelos arcos de conexão, os quais representam as passagens em cada nó, convertendo-se de um link para outro. Estes arcos podem ser penalizados com custos apropriados, de acordo com a inconveniência de cada conversão. Entre eles estão os arcos de conexão que representam os retornos em U. Seja $U_{1R} \subseteq B_1$ o subconjunto de arcos de conexão que referem-se aos retornos em U nos nós requeridos N_{1R}.

Seja D_2 a atual matriz de custos associada a G_2.

Passo 3

Formar o grafo completo $G_3 = (N_{1R}, A_{1R} \cup E_{1R} \cup S_1)$, onde N_{1R} representa os nós requeridos em G_2, e $S_1 \supset B_1$ representa o conjunto de caminhos mínimos entre todos pares de nós $i, j \in N_{1R}$, calculados em G_2, tais que o arco $(i, j) \notin A_{1R} \cup E_{1R}$.

Os arcos de conexão em B_1 estão agora incluídos em S_1. Entretanto, seus custos podem ter sido reduzidos por meio das distâncias mais curtas. Considere D_3 como sendo a matriz de distâncias associada ao grafo completo G_3. Ela reflete, além dos custos dos arcos, todas as penalizações de conversões indesejáveis nos nós.

Passo 4

Construir o grafo transformado final, $G_4 = (N_{2R}, E_{2R} \cup S_2)$ eliminando todos os arcos em G_3 que pertencem a A_{1R}, e todos os arcos de conexão em U_{1R}, que representam os retornos U, unificando seus nós inicial e final, como descrito abaixo.

Para cada arco $(i, j) \in A_{1R}$, identificar os arcos $(i, v) \in S_1$ associados, atribuindo aos mesmos os custos dos arcos $(j, v) \in S_1$ correspondentes. Em seguida eliminar o nó j e todos os arcos nele incidentes.

E para cada arco $(i, j) \in U_{1R}$, identificar os arcos $(v, i) \in S_1$ associados, atribuindo a cada um destes o custo do arco $(v, j) \in S_1$ correspondente. Em seguida eliminar o nó j e

todos os arcos nele incidentes. Os pares de arcos $(i,j),(k,l) \in E_{1R}$, que representam as arestas originais requeridas, continuarão existindo, porém seus nós iniciais devem ser alterados, passando a ser $(l,j),(j,l) \in E_{2R}$.

Estas modificações reduzem o conjunto de nós N_{1R} para N_{2R}, pares de arcos E_{1R} para E_{2R} e o conjunto de caminhos mínimos S_1 para S_2. Observe que cada arco requerido em A' e cada nó requerido em N_X agora estão representados por um único nó, e cada aresta requerida em E' por dois nós conectados por um par de arcos contrariamente orientados, no grafo transformado G_4.

Para completar a transformação, alterar o custo de cada arco em E_{2R} para $-M$, onde M é um número positivo grande.

Desta forma, o Problema Geral de Roteamento com Conversões Penalizadas no grafo misto *G*, se transforma num Problema de Roteamento de Nós em um grafo completamente orientado G_4. A correspondência entre os dois problemas pode ser verificada de uma forma semelhante a que foi feita para a versão PCCM do algoritmo, considerando que cada nó requerido no grafo original é representado por um nó no grafo transformado, cada arco também por um nó, e cada aresta por dois nós interligados por um par de arcos com custos altamente negativos.

O Problema de Roteamento de Nós no G_4 pode ser resolvido com uma rotina de PCVA. Uma solução ótima (ou próximo a ótima) de PCVA em G_4, corresponderá a um circuito ótimo (ou próximo a ótimo) em *G*, o qual cobrirá todos os componentes requeridos com custo total minimizado, não apenas devido às distâncias, mas também devido às penalidades impostas às conversões indesejáveis.

A complexidade do algoritmo apresentado acima, considerando a mesma lógica de complexidade da versão para o PCCM, é de $O\left(\left(2|A|+4|E|\right)^3\right)$, devido ao cálculo de caminhos mínimos no passo 3. Isso indica que a fase de transformação se constitui de um procedimento eficiente, possível de ser resolvido em tempo polinomial. Portanto, a eficiência global do método dependerá da complexidade do algoritmo a ser usado na segunda fase (solução do PCVA).

6.4 Testes Computacionais para o PCRMCP

Os testes computacionais apresentados nessa seção têm como objetivo medir a eficiência do método proposto, comparativamente com alguns métodos existentes. Como já foi visto, um dos trabalhos mais significativos na linha do algoritmo proposto nesta Tese é o de Corberán et al. [Cor02], em que os autores estudam o Problema do Carteiro Rural Misto com Conversões Penalizadas – PCRMCP, e apresentam testes computacionais. Para efeito de uma comparação direta, os testes computacionais do algoritmo proposto foram também concentrados no mesmo tipo de problema estudado pelos referidos autores. Para tanto, foi utilizado o mesmo conjunto de grafos de teste utilizado por Corberán et al. [Cor02], para os quais a solução exata foi buscada (nem sempre com êxito) pelos autores, juntamente com duas soluções heurísticas.

O referido conjunto de teste consiste de 216 grafos mistos aleatoriamente gerados, com $|N| \in [40, 200]$, $|A| \in [90, 440]$, e $|E| \in [10, 40]$. Cada instância foi gerada com os $|N|$ nós dispostos aleatoriamente num reticulado de $[0,100]^2$. Em seguida $|A|$ arcos, $|A| > 2|N|$, foram aleatoriamente gerados, de modo a garantir a conectividade do grafo resultante. Destes, $|A_R|$ arcos foram aleatoriamente escolhidos para serem os arcos requeridos. Finalmente, $|E_R|$ arestas foram acrescentados ao grafo, todos considerados requeridos. Deste modo, as instâncias não possuem arestas não-requeridas, considerando que cada aresta não-requerida pode ser substituída por um par de arcos contrariamente orientados, ambos não-requeridos. Os custos dos links foram fixados como as distâncias euclidianas entre seus respectivos nós terminais.

Mais especificamente na geração de instâncias, para cada par $(|N|, |A|)$ três percentuais de arcos requeridos, 40, 70 e 100% foram considerados. E para cada tripla $(|N|, |A|, |A_R|)$ foram construídos quatro grafos, com acréscimos de 10, 20, 30, e 40 arestas, respectivamente.

Finalmente, foram atribuídos custos de conversão da seguinte forma: 0 para seguir à frente; 1 para conversão à direita; e 3 para conversão à esquerda. Os retornos em U foram considerados proibidos.

A Tabela 6.1 mostra os resultados obtidos dos testes computacionais com estes grafos, juntamente com a solução exata, quando disponível, e as duas soluções heurísticas obtidas por Corberán et al. [Cor02]. Vale ressaltar que as soluções exatas foram buscadas pelos referidos autores para todas as instâncias da Tabela 6.1, usando uma rotina exata de PCV, desenvolvida por Fischetti e Toth [Fis92], como uma sub-rotina de sua solução proposta. Eles estabeleceram um limite de tempo de 28800 segundos para a obtenção da solução ótima, além do qual a tentativa era abandonada. Por conseguinte, dos 216 problemas testados, somente 159 foram resolvidos com otimalidade; para outras 57 instâncias, a solução ótima não foi encontrada dentro do limite de tempo estabelecido.

A Tabela 6.1 está organizada na ordem crescente do número dos arcos e arestas requeridos. Para facilitar a referência, os grafos foram classificados em 6 grandes grupos, designados por D_i, $i = 1, \cdots, 6$, em função do número de arcos requeridos, e cada um destes em 4 sub-grupos D_{ij}, $j = 1, \cdots, 4$, conforme o número de arestas requeridas $(10, 20, 30, 40)$, totalizando assim 24 sub-grupos D_{ij}.

Abaixo, segue uma descrição de cada coluna desta tabela:

INST.:	Nome atribuído a cada instância;
$\|N\|$:	Número de nós;
$\|A\|$	Número de arcos;
$\%A_R$:	Percentual de arcos requeridos;
$\|E_R\|$	Número de arestas requeridas;
C:	Comprimento do Circuito de PCRMCP encontrado pelo método proposto;
T:	Tempo de processamento em Pentium IV 2.0 GHz para o método proposto;
C_E	Comprimento do circuito de PCRMCP, dado pela solução exata;
C/C_E:	Razão entre a solução proposta e a exata;
C_1:	Comprimento do circuito de PCRMCP, dado pela Heurística 1 de Corberán et al. [Cor02];
C/C_1:	Razão entre a solução proposta e a Heurística 1;
C_2:	Comprimento do Circuito de PCRMCP, dado pela Heurística 2 de Corberán et al. [Cor02];

C/C_2: Razão entre a solução proposta e a Heurística 2.

Na Tabela 6.1, além das informações sobre cada instância, algumas informações são calculadas por sub-grupo, para sintetizar a média de ocorrências no mesmo. Por exemplo, no primeiro sub-grupo, D_{11}, formado por 10 instâncias, o tempo médio de processamento foi de 5 segundos; a razão C/C_E foi igual a 1, a qual indica que em todas as 10 instâncias a solução ótima foi encontrada pelo método proposto; os valores de C/C_1=0,9517 e C/C_2=0,9844 indicam que os circuitos encontrados pelo método proposto foram, em média, 4,83% mais curtos do que os obtidos pela heurística 1 e 1,54% mais curtos do os obtidos pela heurística 2 de Corberán et al. [Cor02].

As seguintes observações podem ser destacadas a partir dos testes apresentados na Tabela 6.1:

- Dos 159 problemas com a solução ótima conhecida, o método proposto encontrou a solução ótima em 35 casos;
- Considerando todos os problemas, o desvio médio das soluções obtidas em relação às respectivas soluções ótimas foi de 0,15%;
- O maior desvio registrado em relação à solução ótima (quando encontrada) foi de 0,99%;
- Comparando com a Heurística 1 de Corberán et al. [Cor02], em todas as instâncias testadas o método proposto foi superior, exceto em um único caso de empate;
- Comparando com a Heurística 2 de Corberán et al. [Cor02], dos 216 problemas testados, em 185 o método proposto foi superior, empatando em 11 casos; em apenas 20 ocorrências a Heurística 2 foi superior;
- O maior tempo de processamento foi de 619 segundos, registrado para uma instância do Grupo D_6 , com 410 links requeridos.

Tabela 6.1. Testes computacionais para o Problema de Carteiro Rural Misto com Conversões Penalizadas

INST.	|N|	|A|	$\%A_R$	$\|E_R\|$	C	T	C_E	C/C_E	C_1	C/C_1	C_2	C/C_2
P4941	40	90	40	10	4098	1	4098	1	4329	0,9466	4101	0,9993
P4971	40	90	70	10	4879	2	4879	1	5090	0,9585	4879	1
P4911	40	90	100	10	6679	12	6679	1	6957	0,9600	6765	0,9873
P41141	40	110	40	10	4246	1	4246	1	4327	0,9813	4315	0,9840
P41171	40	110	70	10	6443	3	6443	1	6655	0,9681	6443	1
P61441	60	140	40	10	5569	3	5569	1	5951	0,9358	5573	0,9993
P61641	60	160	40	10	6727	10	6727	1	7038	0,9558	6844	0,9829
P81741	80	170	40	10	7159	13	7159	1	7750	0,9237	7186	0,9962
P82041	80	200	40	10	7852	3	7852	1	8239	0,9530	8277	0,9487
P102241	100	220	40	10	8185	3	8185	1	8763	0,9340	8653	0,9459
D11						**5**		**1**		**0,9517**		**0,9844**
P4942	40	90	40	20	3768	1	3768	1	4164	0,9049	3810	0,9890
P4972	40	90	70	20	5017	3	5017	1	5544	0,9049	5134	0,9772
P4912	40	90	100	20	6436	1	6436	1	6759	0,9522	6541	0,9839
P41142	40	110	40	20	4894	2	4894	1	5480	0,8931	4958	0,9871
P41172	40	110	70	20	6478	18	6478	1	6848	0,9460	6626	0,9777
P61442	60	140	40	20	5941	4	—	—	6284	0,9454	5938	1,0005
P61642	60	160	40	20	6915	15	—	—	7327	0,9438	6932	0,9975
P81742	80	170	40	20	7784	15	—	—	8604	0,9047	8141	0,9561
P82042	80	200	40	20	7913	16	7913	1	8559	0,9245	8014	0,9874
P102242	100	220	40	20	9412	12	9404	1,0009	10095	0,9323	9630	0,9734
D12						**9**		**1.0001**		**0,9252**		**0,9830**
P4943	40	90	40	30	4129	5	4109	1,0049	4724	0,8740	4219	0,9787
P4973	40	90	70	30	6041	7	6027	1,0023	6550	0,9223	6065	0,9960
P4913	40	90	100	30	6909	6	6897	1,0017	7416	0,9316	6931	0,9968
P41143	40	110	40	30	5150	2	5144	1,0012	5987	0,8602	5196	0,9911
P41173	40	110	70	30	7031	6	7016	1,0021	7652	0,9188	7130	0,9861
P61443	60	140	40	30	6234	6	6205	1,0047	7141	0,8730	6250	0,9974
P61643	60	160	40	30	7154	4	7142	1,0017	7861	0,9101	7239	0,9883
P81743	80	170	40	30	7428	7	7420	1,0011	8544	0,8694	7722	0,9619
P82043	80	200	40	30	8696	12	8654	1,0049	9528	0,9127	8838	0,9839
P102243	100	220	40	30	9381	13	9372	1,0010	10743	0,8732	9563	0,9810
D13						**7**		**1,0026**		**0,8945**		**0,9861**

Tabela 6.1. Continuação

INST.	|N|	|A|	$\%A_R$	$\|E_R\|$	C	T	C_E	C/C_E	C_1	C/C_1	C_2	C/C_2
P4944	40	90	40	40	4740	17	—	—	5614	0,8443	4822	0,9830
P4974	40	90	70	40	6153	15	—	—	6793	0,9058	6161	0,9987
P4914	40	90	100	40	7583	14	—	—	8151	0,9303	7644	0,9920
P41144	40	110	40	40	5678	9	5647	1,0055	6648	0,8641	5700	0,9961
P41174	40	110	70	40	7266	16	7222	1,0061	7920	0,9174	7395	0,9826
P61444	60	140	40	40	6511	16	6498	1,0020	7317	0,8898	6527	0,9975
P61644	60	160	40	40	7886	12	7848	1,0048	8645	0,9122	7953	0,9916
P81744	80	170	40	40	7958	16	7948	1,0013	9796	0,8124	8268	0,9625
P82044	80	200	40	40	8361	36	—	—	9560	0,8746	8559	0,9769
P102244	100	220	40	40	9194	14	9165	1,0032	10263	0,8958	9552	0,9625
D14						**17**		**1,0038**		**0,8847**		**0,9843**
P41111	40	110	100	10	8046	4	8046	1	8393	0,9587	8046	1
P61471	60	140	70	10	8508	4	8508	1	8783	0,9687	8518	0,9988
P61671	60	160	70	10	10128	26	10124	1,0004	10436	0,9705	10124	1,0004
P81771	80	170	70	10	9557	10	9557	1	10118	0,9446	9570	0,9986
P102641	100	260	40	10	9841	9	9837	1,0004	10222	0,9627	10010	0,9831
P122641	120	260	40	10	11381	4	11378	1,0003	11910	0,9556	11653	0,9767
P122941	120	290	40	10	11108	5	11108	1	11710	0,9486	11272	0,9855
P143041	140	300	40	10	11565	8	11561	1,0003	12235	0,9452	12095	0,9562
P143341	140	330	40	10	14483	10	14430	1,0037	14874	0,9737	14734	0,9830
P163441	160	340	40	10	13027	11	13019	1,0006	13623	0,9563	13602	0,9577
D21						**9**		**1,0006**		**0,9582**		**0,9840**
P41112	40	110	100	20	8528	25	8520	1,0009	9056	0,9417	8543	0,9982
P61472	60	140	70	20	8752	14	8752	1	9679	0,9042	8752	1
P61672	60	160	70	20	10462	17	10458	1,0004	10924	0,9577	10515	0,9950
P81772	80	170	70	20	9514	21	—	—	10564	0,9006	9748	0,9760
P102642	100	260	40	20	10220	19	10222	1	10865	0,9406	10304	0,9918
P122642	120	260	40	20	11487	11	11480	1,0006	12734	0,9021	11791	0,9742
P122942	120	290	40	20	11619	23	11612	1,0006	12608	0,9216	11688	0,9941
P143042	140	300	40	20	12426	35	12416	1,0008	13335	0,9318	13066	0,9510
P143342	140	330	40	20	13653	35	13642	1,0008	14474	0,9433	13991	0,9758
P163442	160	340	40	20	13473	33	13459	1,0010	14873	0,9059	13890	0,9700
D22						**23**		**1,0005**		**0,9250**		**0,9826**

Tabela 6.1. Continuação

INST.	|N|	|A|	$\%A_R$	$\|E_R\|$	C	T	C_E	C/C_E	C_1	C/C_1	C_2	C/C_2
P41113	40	110	100	30	8675	20	8671	1,0005	9250	0,9378	8742	0,9923
P61473	60	140	70	30	9097	24	9089	1,0009	10269	0,8859	9102	0,9995
P61673	60	160	70	30	10392	13	10371	1,0020	11405	0,9112	10500	0,9897
P81773	80	170	70	30	10758	23	10735	1,0021	12117	0,8878	10780	0,9980
P102643	100	260	40	30	11132	29	11130	1,0002	12341	0,9020	11241	0,9903
P122643	120	260	40	30	10711	16	10700	1,0010	11982	0,8939	10933	0,9797
P122943	120	290	40	30	11701	31	11696	1,0004	13218	0,8852	11940	0,9800
P143043	140	300	40	30	13195	18	13176	1,0014	14603	0,9036	13505	0,9770
P143343	140	330	40	30	14145	25	14113	1,0023	15494	0,9129	14518	0,9743
P163443	160	340	40	30	13489	20	13474	1,0011	14973	0,9009	14065	0,9590
D23						**22**		**1,0012**		**0,9021**		**0,9840**
P41114	40	110	100	40	9151	29	—	—	9995	0,9156	9192	0,9955
P61474	60	140	70	40	9014	28	9000	1,0016	10270	0,8777	9141	0,9861
P61674	60	160	70	40	10880	36	10850	1,0028	11743	0,9265	10873	1,0006
P81774	80	170	70	40	10901	40	—	—	12308	0,8857	11012	0,9899
P102644	100	260	40	40	11328	22	—	—	12601	0,8990	11375	0,9959
P122644	120	260	40	40	11666	32	—	—	13221	0,8824	11807	0,9881
P122944	120	290	40	40	12406	47	12383	1,0019	13561	0,9148	12669	0,9792
P143044	140	300	40	40	13415	52	—	—	15728	0,8529	13673	0,9811
P143344	140	330	40	40	15125	40	—	—	17164	0,8812	15290	0,9892
P163444	160	340	40	40	14222	42	—	—	16125	0,8820	14498	0,9810
D24						**37**		**1,0021**		**0,8918**		**0,9887**
P61411	60	140	100	10	10301	5	10301	1	10807	0,9532	10303	0,9998
P61611	60	160	100	10	13520	15	13520	1	13864	0,9752	13520	1
P81711	80	170	100	10	11494	16	11492	1,0002	11725	0,9803	11502	0,9993
P82071	80	200	70	10	11309	6	11305	1,0004	11711	0,9657	11396	0,9924
P102271	100	220	70	10	11700	9	11700	1	12306	0,9508	11700	1
P163741	160	370	40	10	15010	20	15008	1,0001	15582	0,9633	15438	0,9723
P183841	180	380	40	10	14807	6	14780	1,0018	14957	0,9900	15275	0,9694
P184041	180	400	40	10	15376	23	15371	1,0003	15898	0,9672	15794	0,9735
P204241	200	420	40	10	16308	37	16298	1,0006	17104	0,9535	17060	0,9559
P204441	200	440	40	10	18044	20	18001	1,0024	18592	0,9705	18638	0,9681
D31						**16**		**1,0006**		**0,9670**		**0,9831**

Tabela 6.1. Continuação

INSTÂNCIA	|N|	|A|	$\%A_R$	$\|E_R\|$	C	T	C_E	C/C_E	C_1	C/C_1	C_2	C/C_2
P61412	60	140	100	20	10879	27	10879	1	11307	0,9621	10879	1
P61612	60	160	100	20	13073	36	13073	1	13284	0,9841	13184	0,9916
P81712	80	170	100	20	12365	29	12346	1,0015	13321	0,9282	12449	0,9933
P82072	80	200	70	20	11540	27	11522	1,0016	11962	0,9647	11651	0,9905
P102272	100	220	70	20	12585	37	12577	1,0006	13356	0,9423	13107	0,9602
P163742	160	370	40	20	14742	24	14711	1,0021	15918	0,9261	15269	0,9655
P183842	180	380	40	20	15206	18	15204	1,0001	16633	0,9142	15721	0,9672
P184042	180	400	40	20	15685	28	15679	1,0004	17383	0,9023	16119	0,9731
P204242	200	420	40	20	16491	43	—	—	17380	0,9488	17114	0,9636
P204442	200	440	40	20	18315	66	18299	1,0009	19701	0,9296	18958	0,9661
D32						**34**		**1,0008**		**0,9403**		**0,9771**
P61413	60	140	100	30	10769	59	10721	1,0045	11423	0,9427	10843	0,9932
P61613	60	160	100	30	13104	42	13103	1,0001	13816	0,9485	13227	0,9907
P81713	80	170	100	30	12862	46	—	—	13842	0,9292	12937	0,9942
P82073	80	200	70	30	12282	45	12258	1,0020	13218	0,9292	12356	0,9940
P102273	100	220	70	30	12440	57	12429	1,0009	13473	0,9233	12526	0,9931
P163743	160	370	40	30	14175	34	14036	1,0099	15822	0,8959	14531	0,9755
P183843	180	380	40	30	15445	52	15428	1,0011	17189	0,8985	16024	0,9639
P184043	180	400	40	30	16254	54	16203	1,0031	17839	0,9111	16638	0,9769
P204243	200	420	40	30	16668	51	16625	1,0026	18433	0,9042	17053	0,9774
P204443	200	440	40	30	18378	71	18325	1,0029	20334	0,9038	18936	0,9705
D33						**51**		**1,0030**		**0,9187**		**0,9829**
P61414	60	140	100	40	11604	58	—	—	12761	0,9093	11593	1,0009
P61614	60	160	100	40	13411	69	13391	1,0015	14354	0,9343	13456	0,9967
P81714	80	170	100	40	13334	80	—	—	14552	0,9163	13394	0,9955
P82074	80	200	70	40	11766	30	—	—	12878	0,9137	11805	0,9967
P102274	100	220	70	40	13189	73	—	—	14231	0,9268	13622	0,9682
P163744	160	370	40	40	15321	88	—	—	17040	0,8991	15650	0,9790
P183844	180	380	40	40	16797	86	—	—	18827	0,8922	17225	0,9752
P184044	180	400	40	40	16791	95	—	—	19008	0,8834	17318	0,9696
P204244	200	420	40	40	17364	71	—	—	18899	0,9188	17662	0,9831
P204444	200	440	40	40	19003	87	—	—	21876	0,8687	19393	0,9799
D34						**74**		**1,0015**		**0,9062**		**0,9845**

Tabela 6.1. Continuação

INSTÂNCIA	|N|	|A|	$\%A_R$	$\|E_R\|$	C	T	C_E	C/C_E	C_1	C/C_1	C_2	C/C_2
P82011	80	200	100	10	13691	34	13691	1	13920	0,9835	13691	1
P102211	100	220	100	10	14538	30	14538	1	14969	0,9712	14641	0,9930
P102671	100	260	70	10	14696	46	14696	1	15170	0,9688	14713	0,9988
P122671	120	260	70	10	15416	49	15406	1,0006	16036	0,9613	15440	0,9984
P122971	120	290	70	10	15832	49	15828	1,0003	15968	0,9915	15874	0,9974
P143071	140	300	70	10	16657	64	16657	1	17190	0,9690	16688	0,9981
P143371	140	330	70	10	19795	64	19793	1,0001	20278	0,9762	19814	0,9990
P163471	160	340	70	10	17973	52	17965	1,0004	18332	0,9804	18256	0,9845
P163771	160	370	70	10	21013	29	21011	1,0001	22135	0,9493	21011	1,0001
D41						**46**		**1,0002**		**0,9724**		**0,9966**
P82012	80	200	100	20	14256	62	14240	1,0011	14488	0,9840	14295	0,9973
P102212	100	220	100	20	14695	81	14695	1	15501	0,9480	14835	0,9906
P102672	100	260	70	20	15238	53	15232	1,0004	15997	0,9526	15242	0,9997
P122672	120	260	70	20	14942	50	14933	1,0006	16176	0,9237	14995	0,9965
P122972	120	290	70	20	16485	60	16478	1,0004	17476	0,9433	16627	0,9915
P143072	140	300	70	20	16620	68	16576	1,0027	17457	0,9521	16682	0,9963
P143372	140	330	70	20	20710	85	20686	1,0012	21958	0,9432	20711	1,0000
P163472	160	340	70	20	18634	30	18601	1,0018	19988	0,9323	18623	1,0006
P163772	160	370	70	20	19839	124	19799	1,0020	20597	0,9632	20141	0,9850
D42						**68**		**1,0011**		**0,9491**		**0,9953**
P82013	80	200	100	30	14564	83	14560	1,0003	15246	0,9553	14570	0,9996
P102213	100	220	100	30	15518	85	—	—	16465	0,9425	15580	0,9960
P102673	100	260	70	30	15437	51	15417	1,0013	16623	0,9287	15449	0,9992
P122673	120	260	70	30	15624	72	15596	1,0018	16461	0,9492	15921	0,9813
P122973	120	290	70	30	17006	107	16991	1,0009	17974	0,9461	17248	0,9860
P143073	140	300	70	30	17924	92	17875	1,0027	19644	0,9124	18028	0,9942
P143373	140	330	70	30	20738	97	20671	1,0032	22186	0,9347	20788	0,9976
P163473	160	340	70	30	18402	94	18354	1,0026	20047	0,9179	18529	0,9931
P163773	160	370	70	30	22073	170	22053	1,0009	23178	0,9523	22108	0,9984
D43						**95**		**1,0014**		**0,9377**		**0,9939**

Tabela 6.1. Continuação

INST.	|N|	|A|	$\%A_R$	$\|E_R\|$	C	T	C_E	C/C_E	C_1	C/C_1	C_2	C/C_2
P82014	80	200	100	40	15086	106	—	—	16361	0,9221	15126	0,9974
P102214	100	220	100	40	16081	85	—	—	17423	0,9230	16240	0,9902
P102674	100	260	70	40	14829	54	14783	1,0031	16014	0,9260	14876	0,9968
P122674	120	260	70	40	15975	99	15890	1,0053	17786	0,8982	15942	1,0021
P122974	120	290	70	40	17277	106	17201	1,0044	19353	0,8927	17326	0,9972
P143074	140	300	70	40	17651	88	—	—	19317	0,9138	17959	0,9828
P143374	140	330	70	40	21010	170	20883	1,0061	22972	0,9146	21113	0,9951
P163474	160	340	70	40	19311	171	—	—	22028	0,8767	19588	0,9859
P163774	160	370	70	40	22193	141	—	—	23815	0,9319	22192	1,0000
D44						**113**		**1,0047**		**0,9110**		**0,9942**
P102611	100	260	100	10	18119	67	18119	1	18119	1,0000	18119	1
P122611	120	260	100	10	17613	51	17613	1	18054	0,9756	17641	0,9984
P122911	120	290	100	10	20544	51	20524	1,0010	20812	0,9871	20632	0,9957
P143011	140	300	100	10	19160	40	19156	1,0002	19817	0,9668	19156	1,0002
P183871	180	380	70	10	20418	55	20399	1,0009	21438	0,9524	20493	0,9963
P184071	180	400	70	10	21821	44	21786	1,0016	22525	0,9687	21934	0,9948
P204271	200	420	70	10	22178	117	22169	1,0004	22616	0,9806	22478	0,9867
P204471	200	440	70	10	25290	79	25271	1,0008	26059	0,9705	25359	0,9973
D51						**63**		**1,0006**		**0,9752**		**0,9962**
P102612	100	260	100	20	19308	116	19265	1,0022	19895	0,9705	19306	1,0001
P122612	120	260	100	20	17692	77	17681	1,0006	18571	0,9527	17733	0,9977
P122912	120	290	100	20	20651	161	20544	1,0052	21473	0,9617	20545	1,0052
P143012	140	300	100	20	19805	81	19798	1,0004	20812	0,9516	19798	1,0004
P183872	180	380	70	20	21251	90	21218	1,0016	22749	0,9342	21672	0,9806
P184072	180	400	70	20	21418	98	21390	1,0013	22713	0,9430	21704	0,9868
P204272	200	420	70	20	21649	171	21616	1,0015	22474	0,9633	21668	0,9991
P204472	200	440	70	20	25228	193	25189	1,0015	26280	0,9600	25254	0,9990
D52						**123**		**1,0018**		**0,9546**		**0,9961**

Tabela 6.1. Continuação

INST.	|N|	|A|	$\%A_R$	$\|E_R\|$	C	T	C_E	C/C_E	C_1	C/C_1	C_2	C/C_2
P102613	100	260	100	30	18709	111	18705	1,0002	19628	0,9532	18761	0,9972
P122613	120	260	100	30	18172	143	—	—	18922	0,9604	18127	1,0025
P122913	120	290	100	30	21394	181	21351	1,0020	22517	0,9501	21404	0,9995
P143013	140	300	100	30	20130	159	—	—	21715	0,9270	20130	1
P183873	180	380	70	30	21311	210	—	—	23394	0,9110	21347	0,9983
P184073	180	400	70	30	22789	180	22700	1,0039	24533	0,9289	23091	0,9869
P204273	200	420	70	30	22225	159	—	—	23522	0,9449	22213	1,0005
P204473	200	440	70	30	25631	281	25586	1,0018	27377	0,9362	25672	0,9984
D53						**178**		**1,0020**		**0,9390**		**0,9979**
P102614	100	260	100	40	19204	290	—	—	20359	0,9433	19225	0,9989
P122614	120	260	100	40	18676	255	—	—	19586	0,9535	18677	0,9999
P122914	120	290	100	40	21086	277	—	—	22363	0,9429	21225	0,9935
P143014	140	300	100	40	21008	181	—	—	22951	0,9153	21064	0,9973
P183874	180	380	70	40	21420	182	—	—	23939	0,8948	21530	0,9949
P184074	180	400	70	40	22114	285	—	—	23374	0,9461	22393	0,9875
P204274	200	420	70	40	23002	213	—	—	24612	0,9346	23184	0,9921
P204474	200	440	70	40	26636	419	—	—	29145	0,9139	26672	0,9987
D54						**263**				**0,9306**		**0,9954**
P143311	140	330	100	10	24982	152	24982	1	25560	0,9774	25091	0,9957
P163411	160	340	100	10	20714	90	20696	1,0009	21466	0,9650	20696	1,0009
P163711	160	370	100	10	25722	32	25692	1,0012	26213	0,9813	25775	0,9979
P183811	180	380	100	10	24169	73	24157	1,0005	24946	0,9689	24218	0,9980
P184011	180	400	100	10	26089	111	25997	1,0035	26317	0,9913	26003	1,0033
P204211	200	420	100	10	25321	91	25309	1,0005	26006	0,9737	25393	0,9972
P204411	200	440	100	10	29660	271	29536	1,0042	30054	0,9869	29573	1,0029
D61						**117**		**1,0015**		**0,9778**		**0,9994**

Tabela 6.1. Continuação

INST.	|N|	|A|	$\%A_R$	$\|E_R\|$	C	T	C_E	C/C_E	C_1	C/C_1	C_2	C/C_2
P143312	140	330	100	20	24836	245	24782	1,0022	25848	0,9608	24993	0,9937
P163412	160	340	100	20	21478	170	21395	1,0039	23001	0,9338	21462	1,0007
P163712	160	370	100	20	25589	206	25584	1,0002	26690	0,9587	25736	0,9943
P183812	180	380	100	20	24410	130	24351	1,0024	25510	0,9569	24351	1,0024
P184012	180	400	100	20	26945	241	26935	1,0004	28496	0,9456	26956	0,9996
P204212	200	420	100	20	25480	283	25444	1,0014	27126	0,9393	25444	1,0014
P204412	200	440	100	20	30717	399	30647	1,0023	32400	0,9481	30751	0,9989
D62						**239**		**1,0018**		**0,9490**		**0,9987**
P143313	140	330	100	30	24482	450	24424	1,0024	25535	0,9588	24524	0,9983
P163413	160	340	100	30	22061	328	—	—	23791	0,9273	22146	0,9962
P163713	160	370	100	30	26376	161	—	—	27374	0,9635	26376	1
P183813	180	380	100	30	24916	438	—	—	26262	0,9487	25135	0,9913
P184013	180	400	100	30	26850	244	—	—	28151	0,9538	27006	0,9942
P204213	200	420	100	30	26558	375	—	—	28087	0,9456	26588	0,9989
P204413	200	440	100	30	31166	498	—	—	32800	0,9502	31265	0,9968
D63						**356**		**1,0024**		**0,9497**		**0,9965**
P143314	140	330	100	40	25658	436	—	—	27180	0,9440	25680	0,9991
P163414	160	340	100	40	22374	287	—	—	23913	0,9356	22403	0,9987
P163714	160	370	100	40	27087	619	—	—	28987	0,9345	27148	0,9978
P183814	180	380	100	40	25536	232	—	—	27391	0,9323	25627	0,9964
P184014	180	400	100	40	27194	224	—	—	29147	0,9330	27225	0,9989
P204214	200	420	100	40	26803	375	—	—	28330	0,9461	26891	0,9967
P204414	200	440	100	40	31623	467	—	—	34249	0,9233	31677	0,9983
D64						**377**				**0,9355**		**0,9980**

A Tabela 6.2 apresenta um resumo dos testes computacionais realizados. Abaixo, segue uma descrição de cada coluna desta tabela:

Grupo: Especificação do grupo e subgrupo dos grafos testados, classificados por ordem de número de links requeridos;

Interv.$|A_R|$: Intervalo que contém o número de arcos requeridos do grupo;

$|E_R|$ Número de arestas requeridas do grupo;

Interv.$|N^T|$: Intervalo que contém o número de nós do grafo transformado (dimensão da matriz de distâncias final) para as instâncias do grupo;

No. Inst.: Número de instâncias contidas no sub-grupo;

Ótima Conh.: Número de instâncias do sub-grupo que tem solução ótima conhecida;

Ótima Calcul: Número de instâncias do sub-grupo com solução ótima calculada pelo método proposto;

C/C_E: Razão média entre a solução proposta e a exata;

C/C_1: Razão média entre a solução proposta e a Heurística 1;

C/C_2: Razão média entre a solução proposta e a Heurística 2;

T: Tempo médio de processamento em Pentium IV 2.0 GHz, para o método proposto.

Como pode ser visto na Tabela 6.2, as soluções obtidas pelo método proposto ficaram em média 0,15% acima da solução ótima. Comparadas com as heurísticas 1 e 2 de Corberán et al. [Cor02], as soluções obtidas pelo método proposto foram melhores 6,64% e 1,06%, em média, respectivamente. Quanto ao tempo médio de processamento entre todas as instâncias, as relações são inversas. Enquanto, este tempo médio para o método proposto foi de 86 segundos num Pentium IV-2000, para as heurísticas 1 e 2 foram 3,5 e 16,5 segundos, respectivamente, num Pentium II-350, conforme os testes realizados pelos referidos autores.

O tempo T, relatado nas tabelas 6.1 e 6.2, se refere ao tempo de processamento para a segunda etapa de solução, quando se resolve um Problema de Caixeiro Viajante Assimétrico. O tempo médio de processamento para a primeira etapa da solução, que consiste na transformação do grafo e cálculo da matriz de distâncias, ficou em torno de 1,3 segundos para o conjunto testado, portanto insignificante em relação ao tempo total.

Tabela 6.2. Resumo dos testes computacionais para o Problema de Carteiro Rural Misto com Conversões Penalizadas

Grupo	Interv. $\|A_R\|$	$\|E_R\|$	Interv. $\|N^T\|$	No. Inst	Ótima conhec.	Ótima calcul.	C/C_E	C/C_1	C/C_2	T
D11	[36, 98)	10	[56, 118)	10	10	10	1	0,9517	0,9844	5
D12		20	[76, 138)	10	7	6	1.0001	0,9252	0,9830	9
D13		30	[96, 158)	10	10	—	1,0026	0,8945	0,9861	7
D14		40	[116, 178)	10	6	—	1,0038	0,8847	0,9843	17
D21	[98, 140)	10	[118, 160)	10	10	4	1,0006	0,9582	0,9840	9
D22		20	[138, 180)	10	9	2	1,0005	0,9250	0,9826	23
D23		30	[158, 200)	10	10	—	1,0012	0,9021	0,9840	22
D24		40	[178, 220)	10	3	—	1,0021	0,8918	0,9887	37
D31	[140, 182)	10	[160, 202)	10	10	3	1,0006	0,9670	0,9831	16
D32		20	[180, 222)	10	9	2	1,0008	0,9403	0,9771	34
D33		30	[200, 242)	10	9	—	1,0030	0,9187	0,9829	51
D34		40	[220, 262)	10	1	—	1,0015	0,9062	0,9845	74
D41	[182, 260)	10	[202, 280)	9	9	4	1,0002	0,9724	0,9966	46
D42		20	[222, 300)	9	9	1	1,0011	0,9491	0,9953	68
D43		30	[242, 320)	9	8	—	1,0014	0,9377	0,9939	95
D44		40	[262, 340)	9	4	—	1,0047	0,9110	0,9942	113
D51	[260, 330)	10	[280, 350)	8	8	2	1,0006	0,9752	0,9962	63
D52		20	[300, 370)	8	8	—	1,0018	0,9546	0,9961	123
D53		30	[320, 390)	8	4	—	1,0020	0,9390	0,9979	178
D54		40	[340, 410)	8	—	—	—	0,9306	0,9954	263
D61	[330, 440]	10	[350, 460]	7	7	1	1,0015	0,9778	0,9994	117
D62		20	[370, 480]	7	7	—	1,0018	0,9490	0,9987	239
D63		30	[390, 500]	7	1	—	1,0024	0,9497	0,9965	356
D64		40	[410, 520]	7	—	—	—	0,9355	0,9980	377
Totais				**216**	**159**	**35**				
Média Geral Pond.							**1,0015**	**0,9336**	**0,9894**	**86,00**

6.5 Considerações Sobre o Tempo de Processamento

Na solução de uma instância, uma vez transformado o grafo pelo procedimento anteriormente descrito, as dimensões do grafo original não terão mais uma influência direta sobre a segunda etapa, quando se resolve o Problema de Caixeiro Viajante Assimétrico. Nesta segunda etapa, o tempo de processamento depende, a priori, das dimensões da matriz de distâncias resultante da primeira fase do método proposto. Estas dimensões são determinadas pelo número de componentes requeridos do grafo, e não pelas suas dimensões originais. Por exemplo, como se observa na Tabela 6.1, a instância P163441 (com 160 nós e 340 arcos) teve um tempo de processamento de 11 segundos, enquanto, a instância P81711, embora com dimensões menores (80 nós e 170 arcos), teve um tempo de 16 segundos. Estes tempos, entretanto, são de fato compatíveis com o número de links requeridos em cada caso: 136 arcos requeridos no primeiro caso e 170 arcos requeridos no segundo, e em ambos os casos 10 arestas requeridas.

O Gráfico 6.1 mostra a evolução do tempo de processamento em função do número de arcos e arestas requeridos. O Gráfico 6.2 mostra o mesmo indicador, numa escala logarítmica de tempo. Pode ser observado que, na escala logarítmica, o crescimento de tempo está visivelmente abaixo de um crescimento linear, para todas as curvas. Daí pode-se afirmar, que pelo menos para o universo de grafos testados, o crescimento do tempo de processamento não ocorreu exponencialmente com o crescimento do numero de links requeridos.

Como foi visto, cada arco requerido é representado por um único nó no grafo transformado, e cada aresta, por dois nós. Portanto, seria natural se com aumento de número de arestas o tempo de processamento crescesse numa taxa duas vezes maior do que cresceria com o aumento de número de arcos. Entretanto, um exame simples dos gráficos 6.1 e 6.2 mostra que o tempo de processamento é bem mais sensível em relação ao número de arestas requeridas do que seria esperado pela lógica acima. Este fato foi observado também por Laporte [Lap97] e Corberán et al. [Cor02], para procedimentos semelhantes.

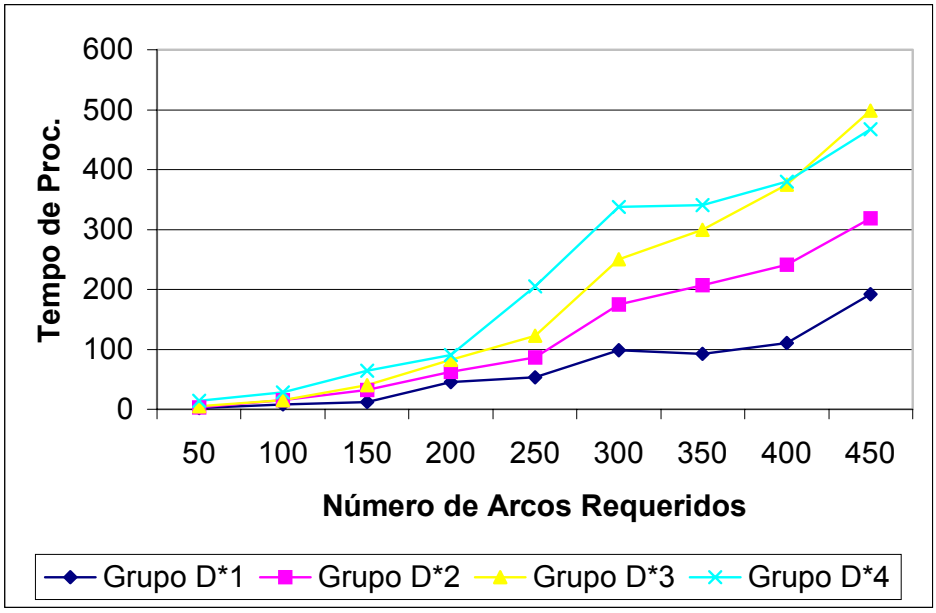


Grafico 6.1 – Evolução do tempo de processamento, em função do número de arcos requeridos.

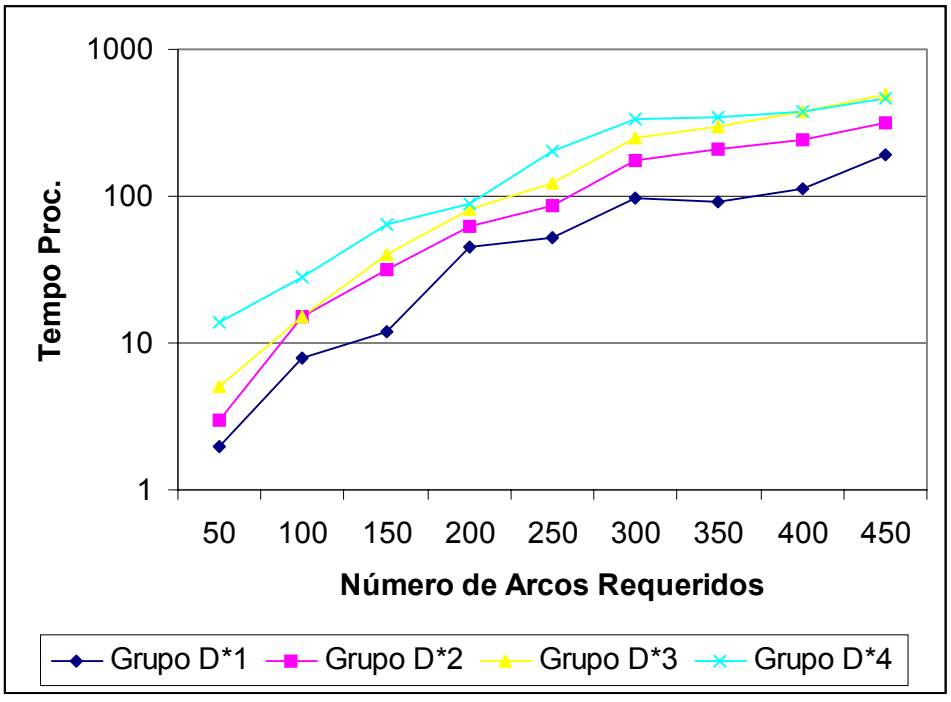


Grafico 6.2 – Evolução do tempo de processamento, na escala logarítmica, em função do número de arcos requeridos.

Diante disso, surge a seguinte pergunta: qual a razão entre as taxas de crescimento do tempo de processamento, quando cresce o número de arestas e quando cresce o número de arcos.

Seja *K* a razão acima definida, e *T* o tempo de processamento para o cálculo de uma instância composta de *a* arcos requeridos e *e* arestas requeridas. Supondo que *T* seja uma função de apenas *a* e *e*, isto é:

$$T = f(a, e) \tag{6.1}$$

então, *K* pode ser definido como:

$$K = \frac{\partial T / \partial e}{\partial T / \partial a} \tag{6.2}$$

O valor de *K* seria um indicador que mostra quão relativamente sensível é o tempo de processamento quanto ao tipo de link. O objetivo é achar uma aproximação numérica para *K*, a partir dos dados disponíveis. Considerando uma aproximação polinomial, tanto o tempo *T*, como o valor de *K* podem ser estimados em função de a e e .

Utilizando os dados contidos na tabela 6.1, inicialmente foi considerada uma aproximação polinomial de quarto grau. Pelos resultados obtidos foi constatado que os termos de primeiro e quarto grau do polinômio tiveram uma participação desprezível na composição do tempo estimado. Portanto, foi dado prosseguimento aos testes de aproximação, considerando o polinômio formado apenas pelos termos de segundo e terceiro grau,

$$\begin{aligned} T &= f(a, e) \\ &= C_1 a^2 + C_2 e^2 + C_3 a e + C_4 a^3 + C_5 e^3 + C_6 a e^2 + C_7 a^2 e \end{aligned} \tag{6.3}$$

onde $C_1, \cdots, C_7$ são parâmetros a serem ajustados a partir dos resultados obtidos. Como se espera uma função monótona e crescente em relação a a e e , devem ser evitados máximos e mínimos locais no conjunto $\{ a \text{ x } e \mid a > 0,\ e > 0 \}$. O que garante essa condição é considerar $C_k \geq 0$ para $\forall\ k = 1, \cdots, 7$.

Para cada instância i dos testes realizados, com a_i arcos requeridos e e_i arestas requeridas, o tempo de processamento T_i é conhecido. Então, os parâmetros de aproximação $C_1, C_2, \cdots, C_7$ podem ser encontrados, minimizando a função *M* abaixo, num procedimento de mínimos quadrados.

$$M = \sum_i (T_i - f(a_i, e_i))^2 .$$

Portanto, considerando o polinômio definido por 6.3, o problema a ser resolvido é:

Minimizar

$$M = \sum_i \left(T_i - C_1 a_i^2 - C_2 e_i^2 - C_3 a_i e_i - C_4 a_i^3 - C_5 e_i^3 - C_6 a_i e_i^2 - C_7 a_i^2 e_i \right)^2 ,$$

sujeito a $C_k \geq 0$, para $\forall\ k\ = 1, \cdots, 7$. (6.4)

Usando os números de arcos e arestas requeridos e os respectivos tempos de processamento fornecidos na tabela 6.1, o problema 6.4 foi resolvido e os seguintes valores foram encontrados para os parâmetros do polinômio:

C_1 = 2,3965 E-5 C_4 = 1,0390 E-6 C_6 = 1,0133 E-4

C_2 = 1,7503 E-6 C_5 = 3,5527 E-21 C_7 = 4,7280 E-5

C_3 = 2,9919 E-4

A figura 6.3 ilustra uma representação gráfica da função 6.3, com parâmetros acima encontrados. Numericamente pode-se conferir, por exemplo, que para a instância P820113, com 200 arcos requeridos e 30 arestas requeridas, o tempo real de processamento foi de 83 segundos, contra a estimativa de 86 segundos encontrada pela aproximação.

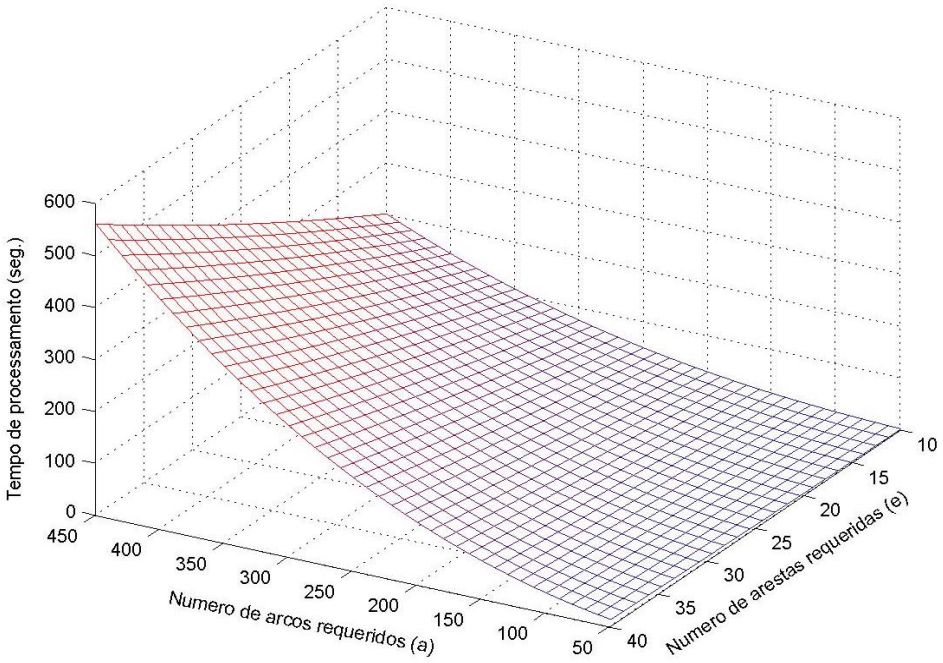


Figura 6.3 Representação gráfica da função aproximada do tempo de processamento.

Calculando as derivadas parciais de T em relação ao número de arcos e arestas requeridas tem-se:

$$\partial T / \partial a = 2C_1 a + C_3 e + 3C_4 a^2 + C_6 e^2 + 2C_7 a e ,$$

$$\partial T / \partial e = 2C_2 e + C_3 a + 3C_5 e^2 + 2C_6 a e + C_7 a^2 .$$

Portanto, a função K definida por 6.3 terá a seguinte forma aproximada:

$$K = \frac{2C_1 a + C_3 e + 3C_4 a^2 + C_6 e^2 + 2C_7 a e}{2C_2 e + C_3 a + 3C_5 e^2 + 2C_6 a e + C_7 a^2} \qquad (6.5)$$

A figura 6.4 mostra a representação gráfica da função K – sensibilidade do tempo de processamento em relação ao número de arestas requeridas.

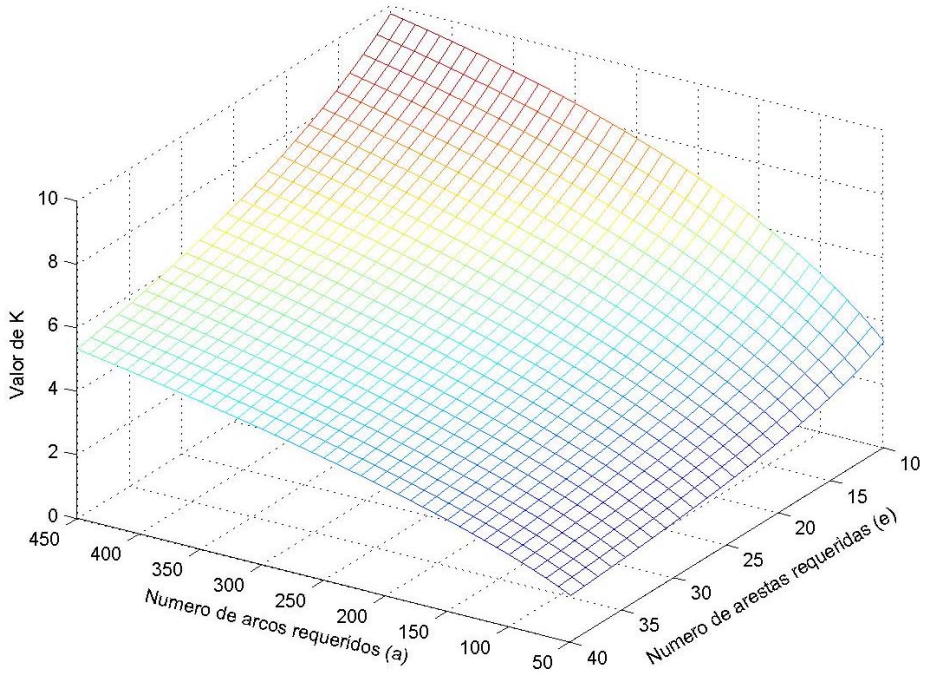


Figura 6.4 Representação gráfica da função K

Exemplificando, para a instância acima referida (P820113, com 200 arcos requeridos e 30 arestas requeridas) é encontrado o valor de K = 3,95. Isto dá uma idéia de que o acréscimo de cada aresta ao problema equivale aproximadamente ao acréscimo de quatro arcos, no aumento de tempo de processamento. Este fato indica que o método proposto, embora aplicável a qualquer instância de PCRMCP, tem maior eficiência, no que se refere

ao tempo de processamento, quando aplicado a grafos que possuem relativamente mais arcos do que arestas. Obviamente esta proporção não é constante, e varia conforme as combinações de a e e.

A figura 6.5 mostra as curvas de níveis de K em função dos números de arcos e arestas requeridos, conforme a expressão 6.5. Pode-se notar que com o aumento de número de arestas requeridas, o valor de K tende a diminuir. Este é um fato positivo registrado por essa análise.

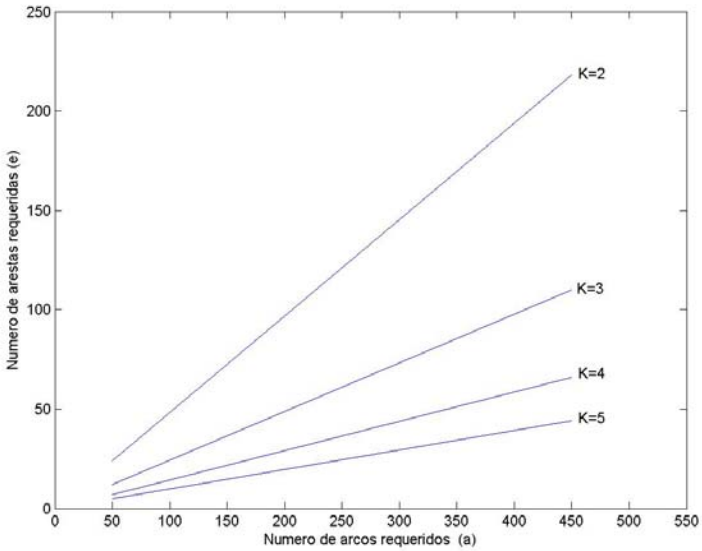


Figura 6.5 Curvas de níveis para a função K

Um outro fato relevante para ser destacado nos testes do Problema de Carteiro Rural Misto com Conversões Penalizadas – PCRMCP em grafos gerais, em comparação com testes realizados para o Problema do Carteiro Chinês Misto – PCCM, em grafos pseudo-manhattan e sem restrições de conversão, é uma marcante diferença entre tamanho de problemas resolvidos em cada caso. No caso de PCCM, foram resolvidos, próximos à solução ótima, problemas com 1000 links (todos obviamente requeridos), num tempo médio de 162 segundos; contra um tempo médio de 272 segundos para PCRMCP, para problemas de aproximadamente 400 links requeridos.

A razão para esta diferença deve ser procurada na estrutura de custos da matriz transformada. A rotina de Busca Local Dirigida, usada para a solução da segunda etapa do problema é bastante sensível à estrutura da matriz de distâncias. As experiências

computacionais mostraram que a rotina usada favorece problemas com uma estrutura de custo mais “discretizado”, como é o caso de grafos manhattan, quando não há grandes variações entre os custos dos links. Grafos gerais, como foi o caso de testes de PCRMCP, têm grandes variações entre os custos de arcos, causando muitas nuances para roteiros a serem formados. Esta característica é agravada com a inserção de penalidades de conversão.

VII – Aplicação do PGR ao Problema de Coleta de Lixo

7.1 Problema de Coleta de Lixo

A coleta de lixo é uma operação logística complexa, realizada quase em todas as zonas urbanas do mundo. Em geral compreende a coleta propriamente dita, compactação, transporte e acondicionamento no aterro sanitário, de grandes quantidades de resíduos diariamente produzidos. Para se ter uma idéia da grandeza do serviço, a Europa produz anualmente 2,5 bilhões de toneladas de lixo sólido [Elk95]. A cidade de São Paulo tem uma coleta diária de mais de 20.000 toneladas de lixo [IBG03].

Pela sua natureza e implicações na qualidade de vida e saúde pública das populações, o serviço deve ser feito assiduamente e dentro de um padrão aceitável de qualidade. Segundo os indicadores sociais do IBGE [IBG03], dos 5.507 municípios brasileiros, 3274 mantém o serviço com freqüência diária. Outros 1104 municípios realizam o serviço três vezes por semana. Dos quase 35 milhões de domicílios localizados nas zonas urbanas do Brasil, 29 milhões são servidos pela coleta na porta; outros 3 milhões se beneficiam com coleta indireta, por meio de caçambas estrategicamente localizadas.

O processo de remoção de lixo compreende em geral as seguintes etapas:

1. a coleta domiciliar, utilizando caminhão coletor, ou outro meio apropriado;
2. transporte até uma estação de transbordo (geralmente disponível nas cidades de pequeno e médio porte), utilizando o mesmo veículo coletor;
3. compactação e / ou outros processamentos, inclusive separação do lixo;
4. transporte da estação para o aterro sanitário, utilizando meios econômicos de transporte; e
5. acondicionamento do lixo no aterro sanitário.

A primeira etapa (coleta) costuma ser a mais dispendiosa, por ser exaustiva e envolver a maior parte de recursos humanos e materiais disponíveis para o serviço. A roteirização de veículos, nessa etapa, é uma aplicação bem conhecida dos Problemas de Roteamento de Arcos. A coleta domiciliar pode ser formulada como PCC, PCCO, PCCM, PCR, PCRO, ou PCRM, a depender da configuração da rede e das exigências específicas do serviço. Entretanto, dependendo de outras restrições que possam existir para cada caso, o problema pode não ter uma formulação padrão, como as de cima.

Seja qual for a formulação, o objetivo é achar um circuito que atenda eficientemente às exigências do serviço. A eficácia do serviço de coleta pode ser expressa em termos da qualidade de coleta em si, da economia total, compreendendo veículos e a mão de obra, e da segurança envolvendo os trabalhadores, a frota e terceiros. Todos esses três parâmetros são sensíveis ao itinerário, seja em função da quilometragem percorrida, ou das manobras e conversões que ele exige.

<table>
<tr><td colspan="6">Eficácia da Coleta</td></tr>
<tr><td>Qualidade</td><td colspan="2">Economia</td><td colspan="3">Segurança</td></tr>
<tr><td>Atendimento à demanda</td><td>Custos associados ao veículo</td><td>Custos associados à mão-de-obra</td><td>Veículo</td><td>Trabalhadores</td><td>Terceiros</td></tr>
<tr><td colspan="6">Todos afetados pelo traçado do percurso da coleta</td></tr>
</table>

Abaixo está uma descrição sucinta de como cada um destes parâmetros de eficiência podem ser positivamente influenciados pelo roteiro. Obviamente existem outros parâmetros, além deste, que também afetam o serviço, porém fora do contexto de um problema combinatorial.

1. A qualidade do serviço pode ser melhorada, uma vez que o roteiro cubra todos os segmentos de rua onde existe lixo para coletar, e corresponda às prioridades de atendimento estabelecidas em cada zona de coleta.

2. A economia associada ao veículo e à mão-de-obra, pode ser considerada minimizando uma função apropriada de custo. A matriz de custos associada à rede deve levar em conta os parâmetros que afetam a economia. O custo associado a cada link l pode ser definido como $C_l = d_l \prod_{lj} p_{lj}$, onde d_l é o comprimento do segmento de rua correspondente, e p_{lj} são as penalidades a serem multiplicadas a este, tais como: penalidades devido ao tipo de pavimentação, a intensidade do trânsito, etc.

3. A segurança pode ser considerada, minimizando os riscos inerentes ao serviço. Um dos riscos é devido às conversões perigosas, que será discutido separadamente na seção 7.2. Outro item de segurança é evitar as passagens ociosas do veículo nas ruas de trânsito intenso, ou locais de grande tráfego de pedestres, o qual pode ser considerado, com a formação apropriada de matriz de custos, acima definida.

Com estas considerações, pode-se achar um roteiro que minimize o custo total da etapa de coleta, incluindo o consumo de combustível, o desgaste do veículo, a duração do serviço (conseqüentemente a incidência de horas-extras) e acidentes de trânsito.

7.2 Algumas Restrições Adicionais do Problema de Coleta de Lixo

Em geral os problemas de roteirização na coleta de lixo envolvem restrições adicionais, a maioria contemplada pelos métodos propostos nos capítulos 5 e 6. Algumas destas restrições são descritas a seguir.

Restrições de Conversão

As restrições de conversão, a exemplo de proibição de retornos U e conversões à esquerda, estão entre as mais relevantes na definição de itinerários para caminhões coletores de lixo, quando o problema é formulado para as zonas urbanas. Como foi visto no capítulo 6, o método proposto assimila tais restrições, incorporando-as no próprio processo de transformação do PRA em PRN.

Entretanto, a potencialidade do método é mais do que o simples atendimento às normas de trânsito; é possível "suavizar" o roteiro gerado, e proporcionar mais segurança para o veículo e terceiros por meio de atribuição apropriada de penalidades também para as conversões permitidas. Em geral, o veículo seguir em frente, num cruzamento, é mais vantajoso do que converter à direita, embora ambos trajetos possam ser permitidos; também, uma conversão com ângulo mais aberto é melhor do que uma com ângulo fechado. De fato, o que torna vantajosa uma conversão em relação à outra é o desgaste imposto ao veículo carregado, e os riscos que cada uma pode lhe proporcionar.

Portanto, há de definir uma matriz de penalidades de conversão $\left[p_{ab}\right]_{a\in L}^{b\in L}$, onde p_{ab} é a penalidade aplicada na passagem do arco $a \in L$ para o arco $b \in L$, onde o nó final de *a* é o nó inicial de *b*. Para efeito de definição da matriz de penalidades, no conjunto de links *L* acima, cada aresta deve ser considerada como um par de arcos contrariamente orientados.

Tipos de Coleta

Um segmento de rua, representado por um arco, ou uma aresta, pode ser servido numa única passagem do veículo coletor. Entretanto existem casos em que os resíduos devem ser

removidos em duas passagens distintas, cada vez num dos lados da via. Este é o caso da coleta nas avenidas de maior movimento. Há uma terceira situação, quando não há coleta para ser efetuada no segmento. Todos estes casos podem ser facilmente considerados pelo método proposto.

Coleta Simples: o segmento de rua deve ser representado por um link requerido;

Coleta Dupla: o segmento deve ser representado por um par de arcos, considerando que se ele é de mão única, os arcos terão o mesmo sentido, e se é de mão dupla, os arcos devem ser contrariamente orientados, e ambos requeridos;

Coleta Inexistente: o segmento deve ser representado por um link não-requerido.

Início e Fim da Coleta

Como foi visto nos capítulos 5 e 6, o método proposto encontra um Circuito de Carteiro que cobre todos os links requeridos do grafo, isto é, partindo de um nó, o circuito termina no mesmo nó. No problema de coleta de lixo não são raras as situações em que a coleta começa num nó, mas deve terminar num outro. Esta variação pode ser resolvida com simples acréscimo de um arco artificial ao grafo que representa a malha viária.

Sejam n_i o nó em que se deve iniciar a coleta, e n_f o nó em que o serviço se encerrar. Cria-se o arco artificial (n_f, n_i) no grafo, considerando-o requerido e fixando seu custo igual a *M*, onde *M* é um valor suficientemente grande. O Circuito de Carteiro obtido neste grafo deve conter o arco (n_f, n_i), porém, o roteiro da coleta será construído de modo a iniciar em n_i e terminar em n_f, ignorando o arco artificial. Logicamente, o valor de M deve ser descontado do comprimento final do roteiro calculado. A razão de fixar este valor bastante grande é para evitar que o arco artificial venha a ser utilizado como uma opção de caminho mínimo no passo 3 do algoritmo proposto.

Vale ressaltar que a incorporação destas variações ao algoritmo proposto capacita-o a resolver instâncias mais genéricas do problema de roteamento. Entre estas, talvez a mais genérica já formulada para os casos não capacitados seja a seguinte:

> Dado um grafo misto $G = (N, L)$, com uma matriz de penalidades associada às conversões nos seus vértices, encontrar o caminho mínimo entre um par de nós

distintos *s* e *t* em *N*, que contenha pelo menos m_a vezes $(m_a \geq 0)$ cada link $a \in L$, e pelo menos uma vez cada nó $n \in N'$ ($N' \subseteq N$).

Hierarquia na Coleta

No serviço de coleta de lixo acontecem situações em que algumas ruas devem ser servidas antes (ou depois) das outras. Por exemplo, numa área mista (comercial / residencial) é preferível que a coleta na área comercial seja feita no horário não-comercial. Isso significa que numa coleta matutina, por exemplo, o serviço na área comercial deva ser feito antes da abertura do comércio, e na hipótese da coleta vespertina, depois do fechamento do mesmo.

Este problema é conhecido na literatura como o *Problema Hierárquico do Carteiro – PHC*, com aplicações inclusive na remoção de neves nas vias públicas. Eiselt et al. [Eis95.1] apresentam uma solução polinomial para uma formulação específica do problema num grafo orientado. O problema é NP-hard para um caso genérico.

O método proposto pode ser adaptado para atender a este requisito adicional do problema de coleta de lixo. Considere que no grafo $G = (N, L)$, o conjunto de links requeridos $L' \subseteq L$ esteja particionado em $\{L_1, \cdots, L_k\}$, e uma relação de ordem $\prec$ seja imposta sobre os elementos da partição, de modo que se $L_p \prec L_q$, então os links em L_p devem ser servidos antes dos links em L_q. Considere ainda que n_i e n_f sejam os nós inicial e final do roteiro a ser construído, e suponha que o arco artificial (n_f, n_i), com custo elevado, como foi descrito acima, esteja presente no grafo *G*. Portanto, considera-se a partição mais ampla $\{L_0, L_1, \cdots, L_k\}$, onde $L_0 = \{(n_f, n_i)\}$. O PHC formulado no grafo *G* consiste em determinar um caminho de custo mínimo, se iniciando em n_i e terminando em n_f, e servindo todos os links requeridos conforme a ordem $L_0 \prec L_1 \prec \cdots \prec L_k$.

Seja $G_4 = (N_{2R}, E_{2R} \cup S_2)$ o grafo transformado final, associado a *G*, conforme obtido pelo algoritmo proposto na seção 6.3. Seja também $\{N_0, N_1, \cdots, N_k\}$ a partição dos nós em N_{2R} associada à partição dos links $\{L_0, L_1, \cdots, L_k\}$ em *L*. A partição dos nós implica que $n \in N_i$, se, e somente se, *n* é um nó do grafo transformado, associado ao link $l \in L_i$ no grafo original.

A adaptação do algoritmo consiste em efetuar alterações adicionais no custo de alguns caminhos mínimos contidos em S_2, de modo a inibir seqüências hierarquicamente indesejáveis de links, conforme a seguir:

Examinar todos os pares de nós $s \in N_p$ e $t \in N_q$:

- se, $p = q$, ou $p = q - 1$, ou $p = q + k$, não faça nenhuma alteração;
- caso contrário, remova o arco (s,t) (atribua custo infinito ao elemento correspondente da matriz de custos).

Desta forma, na matriz de distâncias permanecem apenas os caminhos mínimos que conectam um grupo de links, com outro na hierarquia imediata. Em outro caso, os custos referentes serão infinitos. Com isso, a solução do PCV na segunda fase do método proposto permitirá apenas uma solução hierarquicamente viável, de acordo com a ordem estabelecida. Vale ressaltar que, dados dois grupos quaisquer de links L_p e L_q, este método não impede que alguns links em L_q possam ser utilizados antes dos links de L_p, mesmo que $L_p \prec L_q$. Isso ocorre, quando tais links se apresentem como melhor opção de caminho mínimo, conforme o passo 3 do algoritmo da seção 6.3.

7.3 Experiência de Aplicação a um Bairro de Florianópolis

Características Gerais do Serviço de Coleta de Lixo em Florianópolis

O serviço de limpeza urbana da cidade de Florianópolis é mantido pela Companhia Melhoramentos da Capital – COMCAP, uma empresa de economia mista, cujo acionista majoritário é a Prefeitura Municipal de Florianópolis. A empresa cuida de diversos serviços de limpeza, entre eles, a coleta de lixo, varrição das vias públicas, e limpeza de canais. A coleta de lixo é feita em várias modalidades: coleta de lixo domiciliar, remoção de lixo pesado (móveis e eletrodomésticos inúteis), coleta de lixo seletivo, e a remoção de entulhos.

Entre todos os serviços, o que envolve a maior parte dos recursos da empresa é a coleta de lixo domiciliar. Em geral, este serviço é mantido com a freqüência de três vezes por semana para todos os bairros, exceto para o centro, onde é feito diariamente, e algumas zonas de praia que na temporada passam a ser servidos, também, diariamente.

A cidade é dividida em 60 zonas de coleta, cada uma dimensionada para uma, ou duas viagens de um caminhão coletor. A quantidade de lixo removida varia sazonalmente, e de acordo com os dias da semana. No verão há mais lixo para coletar do que no inverno, e no início da semana mais do que no restante da mesma. A quantidade de lixo coletado, considerando uma média anual, é algo em torno de 400 toneladas por dia.

Os resíduos coletados em cada zona são transportados para a estação de transbordo, localizada no bairro Itacorubi, num percurso médio de 20 km. Na estação, além do processamento do lixo seletivo, é feita a transferência dos resíduos para caminhões de maior porte (figura 7.1). Da estação de transbordo o lixo é transportado para o aterro sanitário, localizado no município de Biguaçu, a uma distância de 40 km.

Em novembro de 2003, o serviço da coleta domiciliar teve uma duração total de 3085 horas-caminhão, num percurso total de 22297 km rodados apenas nas zonas de coleta, excluindo o transporte de lixo. Considerando uma guarnição normal de quatro trabalhadores por caminhão, foram utilizados 12.340 homens-hora no serviço da coleta. Obviamente, os números acima serão maiores, se for considerado o transporte até a estação de transbordo; para atender a este item, os coletores rodaram, ainda, mais 45.810 km durante o referido período.

Figura 7.1 Operação de transferência de lixo na estação de transbordo de Itacorubi

A empresa mantém um controle sobre o percurso dos veículos nas zonas de coleta. Em geral, a seqüência a ser seguida pelo motorista é definida conforme as prioridades de cada zona.

Experiência no Bairro de Canasvieiras

Foi escolhido o bairro de Canasvieiras para realizar uma experiência de geração de roteiro para a coleta de lixo domiciliar, usando o método proposto neste trabalho. A razão da escolha foi o fato do bairro ter sido objeto de um estudo por parte da empresa em 2002, quanto à qualidade e características do itinerário seguido pelo motorista nesse bairro. A figura 7.2 mostra a planta do bairro.

Tendo ruas de mão única e dupla, o bairro foi representado por um grafo misto, com 71 nós, e 115 links, dos quais 27 orientados. Uma vez que alguns segmentos não têm coleta (35 no total), o problema de roteirização deve ser formulado como o de Carteiro Rural. Devem, ainda, ser evitados os retornos U. A versão a ser aplicada, portanto, é a do PCRMCP.

O comprimento de cada segmento de rua, em metros, foi utilizado como o custo do respectivo link. Foram atribuídas penalidades de $P_U = 500$ para os retornos U, e $P_D = 10$, $P_E = 30$ para as conversões a direita e esquerda, respectivamente. Não foi aplicada penalidade somente quando o veículo deve seguir em frente num cruzamento. Vale ressaltar que as conversões à esquerda não são proibidas no bairro; a penalidade aplicada é uma forma de considerar a inconveniência destas conversões e os riscos maiores que elas apresentam.

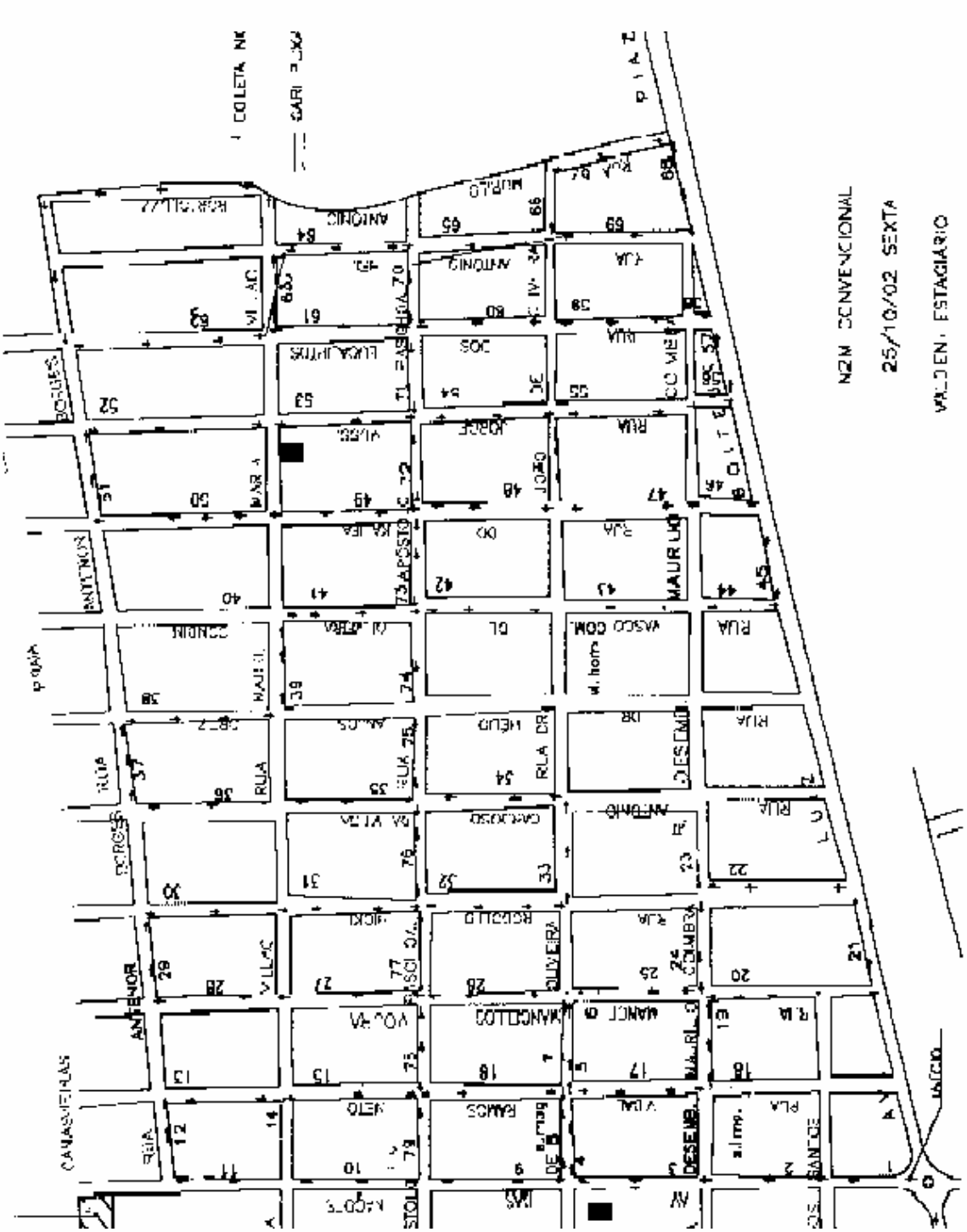


Figura 7.2 Planta do Setor de Coleta N2M – Bairro Canasvieiras, Florianópolis, SC

Como o roteiro do caminhão coletor começa e termina em pontos diferentes, foi utilizado o artifício de inserção de um arco artificial, descrito na seção 7.2.

Com isso, o método proposto no capítulo 6 encontrou um roteiro que minimizou o custo total do percurso, devido às distâncias e conversões. A tabela 7.1 mostra as principais características do roteiro gerado, comparadas com a solução existente.

Como pode ser observado nesta tabela, o roteiro gerado comparado com o existente apresentou melhora significativa na qualidade do roteiro, no que se refere às conversões realizadas, mas não no comprimento do mesmo. De fato, o roteiro existente é mais curto, justamente devido às manobras perigosas realizadas pelo motorista, principalmente os retornos em U. Para evitar tais retornos, certamente há de se modificar completamente o roteiro existente, e incorrer em passagens ociosas adicionais. No cômputo total, considerando os custos acima definidos, o roteiro gerado tem um custo total de 10122, contra o custo de 12101 do roteiro existente. Isto representa uma economia de 16,4% no custo do roteiro.

Tabela 7.1 Características do roteiro gerado, comparadas com o existente.

Parâmetros de avaliação	**Rot. Existente**	**Rot. calculado**
Distância total percorrida (metros)	9371	9432
Distância coletando	8265	8265
Conversões à direita (número)	19	27
Conversões à esquerda	18	14
Retornos U	4	0
Trechos coletando (número)	78	78
Trechos com passagem ociosa (número)	12	14
Trechos com carregamento manual (ruas s/ saída)	2	2
Custo total do roteiro	**12101**	**10122**

Com objetivo de fazer uma comparação mais direta com o roteiro existente, foi feito um outro ensaio, permitindo desta vez que os retornos em U sejam possíveis, e não haja penalidade para as outras formas de conversão. Neste caso, vale como custo do percurso, apenas o comprimento total do roteiro. O resultado deste ensaio, comparado com o roteiro existente, é apresentado na tabela 7.2. Como se vê, o método proposto encontrou um roteiro mais curto, e com menos passagens ociosas, do que o existente.

Tabela 7.2 Roteiro gerado sem penalidades de conversão, comparado com o existente.

Parâmetros de avaliação	**Rot. Existente**	**Rot. calculado**
Distância total percorrida (metros)	9371	9109
Distância coletando	8265	8265
Conversões à direita (número)	19	22
Conversões à esquerda	18	19
Retornos U	4	3
Trechos coletando (número)	78	78
Trechos com passagem ociosa (número)	12	10
Trechos com carregamento manual (ruas s/ saída)	2	2
Custo total do roteiro	**9371**	**9109**

VIII - Conclusões e Recomendações

8.1 Conclusões

Nesse trabalho foi apresentado um método de solução para o Problema Geral de Roteamento que consiste em achar um circuito de custo mínimo que cobre um subconjunto de nós, arcos e arestas de um dado grafo misto. O método proposto oferece uma ferramenta única para resolver uma variedade das instâncias particulares dos problemas de roteamento de arcos, como o Problema do Carteiro Chinês Misto, o Problema do Carteiro Rural e suas variações.

O método consiste em uma transformação do Problema Geral de Roteamento em um problema de roteamento nós. Deste modo, o problema passa a ser a solução de um Problema de Caixeiro Viajante na forma assimétrica. A abordagem é motivada pelos recentes progressos em métodos aproximados para a solução de PCV.

Uma variação de relevância altamente significativa dos problemas de roteamento de arcos é a inserção de restrições de conversão nos nós. Nesse trabalho foi demonstrado que um grafo, mesmo unicursal, pode não ter um circuito euleriano, na presença de tais restrições. A abordagem apresentada é uma das poucas que leva em conta todas as modalidades das restrições de conversão, com aplicação apropriada de penalidade às mesmas.

Os testes computacionais orientados para resolver o Problema do Carteiro Chinês Misto comprovaram a eficiência do método em resolver problemas de porte relativamente grande, com obtenção de soluções de boa qualidade, comparadas com os limites inferiores calculados para cada caso, em tempo aceitável. Em comparação com os resultados computacionais relatados para os algoritmos Mixed, na amostra testada foi verificada a superioridade do método proposto, no que se refere ao desvio médio do limite inferior e nos piores casos atingidos por cada abordagem.

Os testes computacionais, para resolver o Problema do Carteiro Rural Misto com Conversões Penalizadas, foram realizados num conjunto de grafos testados por outros autores na solução do mesmo problema. Os testes foram igualmente satisfatórios, obtendo

soluções de boa qualidade, quando comparadas com soluções exatas e aproximadas disponíveis para cada caso. Das 159 instâncias de teste com a solução exata conhecida, em 35 a solução ótima foi encontrada. O desvio médio em relação à solução ótima, considerando todos os casos testados, foi de 0,15%. As soluções obtidas, todas em tempos aceitáveis de processamento, foram melhores do que os resultados publicados para algumas heurísticas encontradas na literatura.

Os testes num problema real de coleta de lixo mostraram a potencialidade do método em resolver problemas práticos, não apenas na sua forma clássica, mas também na forma real, envolvendo algumas restrições adicionais.

Considerando:

- a qualidade de soluções obtidas, em geral ótimas, ou próximas às ótimas;
- o tempo aceitável de processamento;
- o fato de oferecer uma ferramenta única para uma variedade de problemas de roteamento;
- a incorporação de restrições de conversão nos vértices;
- a possibilidade de inserção de algumas outras restrições comuns a estes problemas;

o método proposto pode ser empregado com vantagem na solução dos problemas de roteamento de arcos.

8.2 Limitações

Como foi visto, a complexidade do procedimento de transformação é de $O\left(\left(2|A|+4|E|\right)^3\right)$ para o caso de PGR. Isso significa que a primeira fase do método é constituída de um algoritmo eficiente. Logo, a eficiência global do método dependerá apenas da segunda fase, compreendida por alguma rotina para a solução do PCV. Os algoritmos exatos disponíveis para este último costumam ter uma complexidade exponencial, razão pela qual não se recomenda o seu emprego. Todavia, o método Busca Local Dirigida, adotada nos testes computacionais, mostrou-se bastante adequado para essa finalidade.

Como foi visto, a dimensão da matriz transformada final é uma função do número de links requeridos do grafo original. Esta dimensão pode crescer muito quando o grafo é denso e tem um grande número de links requeridos. Por isso, o método é indicado para os grafos esparsos (por exemplo, do tipo Manhattan), e não para os grafos densos. Todavia, isto não se constitui em uma limitação forte nos problemas práticos. Em geral, os problemas reais de distribuição são definidos em malhas urbanas ou rodoviárias, representadas por grafos esparsos.

8.3 Recomendações para a Continuidade do Trabalho

Na continuidade deste trabalho é altamente recomendável um estudo sobre a adaptação do procedimento de Busca Local Dirigida, visando a solução do PCV no grafo transformado pelo método proposto. O objetivo seria ajustar o procedimento de modo a explorar a estrutura do grafo transformado, ao invés de resolver um PCV genérico.

IX - Referências

[App95] Applegate D., Bixby R. E., Chvátal V., Cook W., Finding Cuts in the TSP: a Preliminary Report, report 95-05, dimacs, Rutgers University, New Brunswick, NJ, 1995.

[Bar90] Barahona F., On Some Applications of the Chinese Postman Problem. Algorithms and Combinatorics, Vol 9, Path, Flows VLSI Layout, B. Korte et al. eds. Springer-Verlag, Berlin, 1990.

[Bel74] Beltrami E. J., Bodin L. D., Networks and Vehicle Routing for Municipal Waste Collection. Networks 4, 65-94, 1974.

[Ben03.1] Benavent E., Carrotta A., Corberán A., Sanchis J.M., e Vigo D., Lower Bounds and Heuristics for the Windy Rural Postman Problem, Artigo de trabalho, Universidade de Valência, Espanha, Março de 2003.

[Ben03.2] Benavent E., Corberán A., Piñana E., Plana I., e Sanchis J.M. New Heuristic Algorithms for the Windy Rural Postman Problem, Artigo de trabalho, Universidade de Valência, Espanha, Maio de 2003.

[Ber96] Van Den Berg, J. P., Multiple Order Pick Sequencing in a carousel System: A Solvable Case of rural Postman Problem, Journal of Operational Research Society, Vol.47, pag: 1504-1515, 1996.

[Bod78] Lawrence D. Bodin, Samuel J. Kursh, A Computer-Assisted System for the Routing and Scheduling of Street Sweepers, Operations Research, Vol. 26, No. 4, 1978

[Bod79] Lawrence D. Bodin, Samuel J. Kursh, A Detailed description of A Computer System For the Routing and Scheduling of Street Sweepers, Comput. Ops Res. Vol.6, 1979

[Bod83] Bodin L., Golden B., Assad A., Ball M., Routing and Scheduling of Vehicles and Crews – The State of the Art., Computers & Operations Research, Vol.10, No. 2, 1983.

[BTS03] Bureau of Transportation Statistics – BTS, www.bts.gov .

[Car80] Carpaneto G., e Toth P., Some New Branching and Bounding Criteria for the Asymmetric Traveling Salesman Problem. *Management Science*, 26, 736-743, 1980.

[Cor00] Corberan, A. , Marti, R. , Romero, A. Heuristics for the Mixed Rural Postman Problem, Computers & Operations Research, Vol.27, No. 2, pag: 183-203, 2000.

[Cor02] Corberán A., Martí R., Martinez E., e Soler D., The Rural Postman Problem on Mixed Graphs with Turn Penalties, Computers & Operations Research Vol.29, pag. 887 - 903, 2002.

[Cor94] Corberan, A. , Sanchis, J. M., A Polyhedral Approach to the Rural Postman Problem, , European Journal of Operational Research, Vol.79, pag: 95-114, 1994.

[Cor98] Corberan, A. , Sanchis, J. M., The General Routing Problem Polyhedron: Facets from the RPP and GTSP, European Journal of Operational Research, Vol.108, pag: 538-550, 1998.

[Cri75] Christofides N., Graph Theory – An Algorithmic Approach. Academic Press, London, 1975.

[Cri81] Christofides N., Campos V., Corberán A. e Mota E. An Algorithm for the Rural Postman Problem. Imperial College Report IC.ºR.81.5, London, 1981.

[Cri84] Christofides N., Benavent E., Campos V., Corberán A. e Mota E. An Optimal method for the Mixed Postman Problem. In: Thoft-Christensen P, editor. System Modeling and Optimization, Lecture Notes in Control and Information Sciences, vol. 59. Berlin: Springer, 1984.

[Cri86] Christofides N., Campos V., Corberán A. e Mota E. An Algorithm for the Rural Postman Problem on a Directed Graph. Math. Programming Study. 26, 155-166, 1986.

[Dro97] Dror, M., Langevin, A. , A Generalized Traveling Salesman Problem Approach to the Directed Clu stered Rural Postman Problem, Transportation Science, Vol.31, No. 2 pag: 187-192, 1997.

[Edm73] Edmonds, J. and Johnson, E. L., Matching, Euler Tours and the Chinese Postman Problem, Math. Program. 5, Pag: 88-124, 1973.

[Eis95.1] Eiselt H A, Gendreau M, Laporte G, Arc Routing Problems, Part I: The Chinese Postman Problem, Operations Research, Vol. 43, No. 2, 1995

[Eis95.2] Eiselt H A, Gendreau M, Laporte G, Arc Routing Problems, Part II: The Rural Postman Problem, Operations Research, Vol. 43, No. 3, 1995

[Elk95] Elkington J., Can big business help? Peaple & Planet, Vol. 4, no. 1, 1995.

[Eul36] Euler L. 1736. Solutio Problematis ad Geometrian Situs Pertinentis. Commentarii academiae scientarum petropolitanae 8, 128-140.

[Fis92] Fischetti M, Toth P. Na Additive branching procedure for the asymmetric travelling salesman problem. Mathematical Programming, 53: 173-197, 1992.

[For62] Ford L. R., Fulkerson D. R. Flows in Networks. Princeton University Press, Princeton, NJ, 1962.

[Fre78] Frederickson G. N., Hecht M. S., e Kim C. E., Approximation Algorithms for Some postman Problems. *SIAM J. Comput.* 7, 178-193, 1978.

[Fre79] Frederickson, G. N., Approximation Algorithms for Some Postman Problems, J. Assoc. Comput. Mach. 26, pag: 538-554, 1979.

[Gei00] GEIPOT, www.geipot.gov.br .

[Geo74] Geoffrion A. e Graves G, multicommodity distribution system design by Benders decomposition. Management Sci. 20, pag: 822-844, 1974.

[Gia00] Ghiani, G., Importa, G., An Efficient Transformation of the Generalized Vehicle Routing Problem, European Journal of Operational Research, Vol.122, pag: 11-17, 2000.

[Glo00] Glover F., Laguna M. e Martí R., Fundamentals of Scatter Search and Path Relinking, *Control and Cybernetics*, 29, 653-684, 2000.

[Glo97] Glover F., Laguna M. Tabu Search. Dordrecht; Kluwer Academic, 1997.

[Gol83] B. L. Golden, J. S. Dearmon, Computational Experiments with Algorithms for a Class of Routing Problems, Computers Ops Res. Vol.14, No. 1, 1983

[Gro92] Grotschel, M., Win, Z., A Cutting plane Algorithm for the Windy Postman Problem, Mathematical Programming, Vol. 55, pag: 339-358, 1992.

[Gua62] Guan M K, Graphic programming using odd or even points. Chinese Math. 1, 273-277, 1962.

[Gua84] Guan M. K. On the Windy Postman Problem. *Discrete Appl. math*. 9, 41-46, 1984.

[Ham97] Herbert Hamers, Theory and Methodology On the Concavity of delivery Games, European Journal of Operational Research 99, 1997

[IBG03] Instituto Brasileiro de Geografia e Estatística – IBGE, www.gov.br .

[Ita81] Itai A., Lipton R. J., Papadimitriou C. H., Rodeh M., Covering Graphs by Simple Circuits. *SIAM J. Comp*. 10, 746-750, 1981.

[Joh97] Johnson, D. S., McGeoch, L. A., The Traveling Salesman Problem: a Case Study, Aarts, E. & Lenstra, J. K. (eds). Local Search in Combinatorial Optimization, Wiley, New York, pag: 215-310, 1997.

[Kap79] Kappauf C. H. e Koehler G. J., The Mixed Postman Problem. *Discr. Appl. Math*. 1, 89-103, 1979.

[Kar72] Karp R., Reducibility Among Combinatorial Problems. Complexity of Computer Computations, Edited by R. Miller and J. Thatcher, 85-104, *Plenum Press*, New York, 1972.

[Kar79] Karp R. M., A patching Algorithm for the Nonsymetric Traveling salesman Problem. *SIAM Journal on Computing*, 8, 561-573, 1979.

[Kes87] Kesel'man D. Y., Covering the Edges of a Graph by Circuits. *Kibernética* 3, 16-22, 1987.

[Lap97] Laporte, G. , Modeling and Solving Several Classes of Arc Routing Problems as Traveling Salesman Problems, Computers & Operations Research, Vol.24, No. 11, 1997

[Law76] Lawler, E, Combinatorial Optimization, Networks and Matroids, Holt, Rinehart and Winston, 1976.

[Let96] Letchford, A. N., New Inequalities for the General Routing Problem, European Journal of Operational Research, Vol.96, pag: 317-322, 1996.

[Let98] Letchford, A. N., Eglese, R. W., The Rural postman Problem With Deadline Classes, European Journal of Operational Research, Vol.105, pag: 390-400, 1998.

[Let99] Letchford, A. N., A General Routing Polyhedron: A Unifying Framework, European Journal of Operational Research, Vol.112, pag: 122-133, 1999.

[Li 96] Li, L. Y. O., Eglese, R. W., An Interactive Algorithm for Vehicle Routing for Winter-Gritting, Journal of Operational Research Society, Vol.47, pag: 217-228, 1996.

[Lin88] Lin Y, Zhao Y, A New Algorithm for the Directed Chinese Postman Problem, Comput.Opns. Res. Vol.15, No. 6, 1988

[Mal89] Malek M., Mourad A., Pandya M., Topological Testing. Proceedings of the IEEE 1989 International Test Conference, paper 4.4, 103-110, 1989.

[Mal93] Malandraki, C., Daskin, M. S., The Maximum Benefit Chinese Postman Problem and the Maximum Benefit Traveling Salesman Problem, European Journal of Operational Research, Vol.65, pag: 218-234, 1993.

[McB82] Richard McBride, Controlling Left and U-Turns in the Routing of Refuse Collection Vehicles, Computers Ops Res. Vol.9, No. 2, 1982

[Min79] Edward Minieka, The Chinese Postman Problem for Mixed Networks, Management Sciences, Vol 25, No. 7, 1979.

[Mou00] Cândida Mourão, M., Teresa Almeida, M., Lower-Bounding and heuristic methods for a refuse collection vehicle routing problem, European Journal of Operational Research, Vol.121, pag: 420-434, 2000.

[Nob96] Yves Nobert, Jean-Claude Picard, An Optimal Algorithm for the Mixed Chinese Postman Problem, Networks, Vol. 27, 1996

[Noo93] Noon C. E. e Bean J. C., An Efficient Transformation of the Generalized Traveling Salesman Problem. *INFOR*, 31-44, 1993.

[Orl74] Orloff C. S., A Fundamental Problem in Vehicle Routing. Networks 4, 35-64, 1974.

[Pap76] Papadimitriou, C. H., On the Complexity of the Edge Traversing, J. Assoc. Comput. Mach. 23, pag: 544-554, 1976.

[Pea87] Wen Lea Pearn, Transforming Arc Routing Into Node Routing Problems, Computers Ops Res. Vol.14, No. 4, 1987

[Pea89] Wen Lea Pearn, Approximate Solutions for the Capacitated Arc Routing Problem, Computers Ops Res. Vol.16, No. 6, 1989

[Pea91] Wen Lea Pearn, Augment-Insert Algorithms for the Capacitated Arc Routing Problem, Computers Ops Res. Vol.18, No. 2, 1991

[Pea94] Pearn W L, Li M L, Algorithms for the Windy Postman Problem, Computers Ops Res. Vol.21, No. 6, 1994

[Pea95.1] Wen Lea Pearn, C. M. Liu, Algorithms for the Chinese Postman Problem, On Mixed Networks, Computers Ops Res. Vol.22, No. 5, 1995

[Pea95.2] Pearn, W. L., Wu, T. C., Algorithms For The Rural Postman Problem, Computers & Operations Research, Vol.22, No. 8, pag: 819-828, 1995.

[Pea99] Pearn, W. L., Chou, J. B., Improved Solutions For The Chinese Postman Problem on Mixed Network, Computers & Operations Research, Vol.26, No. 8, pag: 819-827, 1999

[Pet77] J. P. Petersen, A Manual Procedure for Designing Mailman-Routes, Comput. Ops. Res.Vol.4, 1977

[Rag99] Raghavachari B., Veerasamy J., A 3/2-Aproximation Algorithm For the Mixed Postman Problem, SIAM J. Discrete Math., Vol 12-4, Pag. 425-433, 1999.

[Ral93] Ralph T. K., on the Mixed Chinese Postman Problem. *Operations Research Letters*, 14, 123-127, 1993.

[Ric91] Richey M B, Parker R G, A cubic algorithm for the directed Eulerian sub graph problem, European Journal of Operational Research 50, 1991

[Rod00] Rodrigues, M. A. P., Problema do Caixeiro Viajante, Um Algoritmo para Resolução de Problemas de Grande Porte Baseado em Busca Local Dirigida – Dissertação de Mestrado, EPS, Universidade Federal de Santa Catarina, Brasil, 2000.

[She88] Sherafat, H. Uma Solução para o Problema do Carteiro Chinês no Grafo Misto, Anais de XXI SBPO, Vol.1, pag: 157-170, 1988.

[She96] Sherafat, H., Circuitos Eulerianos Sujeitos a Restrições nos Vértices, Anais de XXVIII SBPO, Vol.2, pag: 575-580, 1996.

[Ste79] Helman I. Stern, Moshe Dror, Routing Electric Meter Readers, Comput. Ops Res. Vol.6, 1979

[Vou99] Voudouris, C., Tsang, E. Guided Local Search and its Application to the Traveling Salesman Problem, European Journal of Operations Research 113, pag: 469-499, 1999.

[Win89] Win Z, On the Windy Postman Problem on Eulerian Graphs, Mathematical Programming 44, 1989

Anexo I

Algoritmo de Busca Local Dirigida

Anexo I – Algoritmo de Busca Local Dirigida

(Versão editada por Marco Antônio P. Rodrigues [Rod00])

Sejam,

S: um problema de otimização combinatorial;

g: a função objetivo associada;

I: uma função indicadora de uma característica f_i de uma solução, sendo

$$I_i(s) = \begin{cases} 1 \text{ se a solução tem a propriedade } f_i \\ 0 \text{ caso contrário} \end{cases} \quad , s \in S.$$

M: o número de características definidas sobre as soluções;

c_i: o custo associado a cada característica f_i;

P_i: um parâmetro de penalidade correspondente a característica f_i;

h: função de custo aumentada;

λ: um parâmetro de controle sobre a intensidade das restrições na função de custo aumentada; e

Util: uma função de utilidade definida sobre as características de uma solução.

O algoritmo básico para a *Busca Local Dirigida* – BLD pode ser descrito como:

Procedimento BLD(S, g, λ, [I_1,..., I_M], [c_1,..., c_M], M)

Início

$k \leftarrow 0$;

$s_0 \leftarrow$ solução inicial gerada por alguma heurística ou aleatoriamente;

Para $i \leftarrow 1$ até M **faça** $p_i \leftarrow 0$;

Enquanto *critério de parada* **faça**

$h \leftarrow g + \lambda \sum_{i=1}^{M} p_i . I_i$;

$s_{k+1} \leftarrow BuscaLocal(s_k, h)$;

Para $i \leftarrow 1$ até M **faça** $util_i = I_i(s_{k+1}).c_i /(1+p_i)$;

Para cada i tal que $util_i$ é máximo **faça** $p_i \leftarrow p_i+1$;

$k \leftarrow k+1$;

FimEnquanto;

$s^* \leftarrow$ a melhor solução encontrada com respeito à função custo g;

Retorne s^* ;

Fim.

A chamada ao procedimento de *BuscaLocal*() no algoritmo acima pode ser preenchida por qualquer procedimento do gênero. No contexto do Problema de Caixeiro Viajante Assimétrico a busca deve ser do tipo *3-opt*. Para este problema específico Voudouris e Tsang [Vou99] recomendam a utilização de estratégias de redução da vizinhança, num procedimento denominado *Busca Local Rápida* – BLR. O conjunto de vizinhança é redefinido dinamicamente, utilizando uma variável binária (*bit*) associada a cada vizinho. O algoritmo de BLD, combinado ao BLR pode ser descrito como:

Procedimento BLD_BLR(S, g, λ, [I_1,..., I_M], [c_1,..., c_M], M, L)
Início
  $k \leftarrow 0$;
  $s_0 \leftarrow$ solução inicial gerada por alguma heurística ou aleatoriamente;
  Para $i \leftarrow 1$ até M **faça** $p_i \leftarrow 0$; *{Faz todas as penalidades iguais a zero}*
  Para $i \leftarrow 1$ até L **faça** $bit_i \leftarrow 1$; *{Ativa todas as sub-vizinhanças}*
  Enquanto *critério de parada* **faça**
    $h \leftarrow g + \lambda \sum_{i=1}^{M} p_i . I_i$;
    $s_{k+1} \leftarrow BLR(s_k, h, [bit_1,...,bit_L], L)$;
    Para $i \leftarrow 1$ **até** M **faça** $util_i = I_i(s_{k+1}).c_i/(1+p_i)$;
    Para cada i tal que $util_i$ é máximo **faça**
      $p_i \leftarrow p_i + 1$;
      $SetBits \leftarrow$ **SubVizinhançasCaracterística**(i);
      {Ativa todas as sub-vizinhanças relativas à característica penalizada}
      Para cada bit b **em** SetBits **faça** $b \leftarrow 1$;
    FimPara;
    $k \leftarrow k+1$;
  FimEnquanto;
  $s^* \leftarrow$ a melhor solução encontrada com respeito à função custo g;

```
   Retorne s* ;
Fim.


Procedimento BLR (S, h, [bit_1,..., bit_L], L)
Início
   Enquanto ∃bit, bit=1 faça
      Para i ←1 até L faça
         Se bit_i=1 então              {Busca de sub-
         vizinhanças para mov. de melhora}
            Mov ← conjunto de movimentos na sub-vizinhança de
            i
            Para cada movimento m em Mov faça
               s' ← m(s);      {s' é a solução gerada pelo
               movimento m}
               Se h(s')<h(s) então
                  bit_i ←1;
                  SetBits ← SubVizinhançasParaMovimento(m);
                  {Ativação de outras sub-vizinhanças}
                  Para cada bit b em SetBits faça b ←1;
                  s' ← s;
                  Vá para MovimentoDeMelhoraEncontrada;
               FimSe;
            FimPara;
            {Desativa uma sub-vizinhança}
            bit_i ← 0;
         FimSe;
         MovimentoDeMelhoraEncontrada: Continue;
      FimPara;
   FimEnquanto;
   Retorna s;
Fim.
```

SubVizinhançasParaMovimento *(m)*

Procedimento que retorna os bits da sub-vizinhança para propagação da ativação quando o movimento *m* é realizado;

SubVizinhançasCaracterística (*i*)
Procedimento que retorna os bits da sub-vizinhança correspondentes a característica *i*;

Nesses procedimentos, cada vértice é visitado na ordem do trajeto, sendo cada um dos arcos adjacentes examinados quanto a uma possível troca que melhore a solução corrente. Na aplicação da *BLD* com a *BLR* o procedimento é semelhante. Inicialmente, todas as sub-vizinhanças estão ativas. Cada sub-vizinhança é examinada em uma ordem arbitrária estática (por exemplo, da primeira a *n*-ésima cidade). Cada vez que uma sub-vizinhança ativa é encontrada, tenta-se encontrar um movimento de melhora *3-opt*. Se não for encontrado um movimento, a sub-vizinhança é feita inativa (o bit correspondente é ajustado para 0). Caso contrário, o primeiro movimento encontrado é realizado e as sub-vizinhanças correspondentes às cidades no fim de cada arco envolvido (removido ou adicionado pelo movimento) são ativadas (os bits correspondentes são ajustados para 1). O processo sempre continua com a próxima sub-vizinhança na ordem estática. Sempre que uma volta completa por todas as sub-vizinhanças ocorra, sem que qualquer movimento de melhora seja encontrado, o processo termina e retorna o trajeto encontrado.

Completando a descrição dos elementos necessários a aplicação da *BLD*, definem-se o conjunto de *características de uma solução* como todos os arcos e_{ij} do grafo que representa o PCV; o *custo da característica* como o comprimento de cada arco, d_{ij}; e a função *indicadora da característica* por

$\acute{U}til\ (trajeto,\ e_{ij}) = Ie_{ij}\ (trajeto).\ d_{ij}/(1+ p_i)$,

onde $Ie_{ij}\left(trajeto\right) = \begin{cases} 1\ ;\ \text{se } e_{ij}\ \text{pertence ao trajeto} \\ 0\ ;\ \text{caso contrário} \end{cases}$.

Anexo II

Código em Object Pascal (Delphi 4.0) do algoritmo proposto

Anexo II – Código em Object Pascal do Algoritmo Proposto

```
// URandGraf gera um grafo aleatório, na forma pseudo-Manhattan, com uma solução
// próxima a ótima de MCPP conhecida.
// Versão 3, Setembro de 2002

unit URandGraf;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ComCtrls, ExtCtrls, UPrincipal, Buttons, Spin;

type
  TForm1 = class(TForm)
    Button1: TButton;
    StatusBar1: TStatusBar;
    ListBox1: TListBox;
    Button2: TButton;
    GroupBox1: TGroupBox;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    EDit4: TEdit;
    Label4: TLabel;
    Edit5: TEdit;
    Label6: TLabel;
    Edit6: TEdit;
    Label7: TLabel;
    Label5: TLabel;
    Label12: TLabel;
    SaveDialog1: TSaveDialog;
    RadioGroup1: TRadioGroup;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    RadioButton3: TRadioButton;
    Button3: TButton;
    GroupBox2: TGroupBox;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    ProgressBar1: TProgressBar;
    Button4: TButton;
    Button5: TButton;
    Label8: TLabel;
    Label13: TLabel;
    Edit7: TEdit;
    GroupBox3: TGroupBox;
    Label15: TLabel;
    Button6: TButton;
    Button7: TButton;
    BitBtn1: TBitBtn;
    Button8: TButton;
    GroupBox4: TGroupBox;
    Edit8: TEdit;
    Label14: TLabel;
    CheckBox1: TCheckBox;
    CheckBox2: TCheckBox;
    CheckBox3: TCheckBox;
    Edit9: TEdit;
    Label16: TLabel;
```

```
    Label18: TLabel;
    GroupBox5: TGroupBox;
    SpinEdit1: TSpinEdit;
    Label19: TLabel;
    CheckBox5: TCheckBox;
    Visualizar: TButton;
    Button9: TButton;
    Edit11: TEdit;
    Edit12: TEdit;
    Label20: TLabel;
    Label21: TLabel;
    GroupBox6: TGroupBox;
    Edit13: TEdit;
    Edit14: TEdit;
    Edit15: TEdit;
    Edit10: TEdit;
    Label17: TLabel;
    Label22: TLabel;
    Label23: TLabel;
    Label24: TLabel;
    Label25: TLabel;
    CheckBox4: TCheckBox;
    Edit16: TEdit;
    Label26: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure RadioGroup1Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
    procedure Button7Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure Button8Click(Sender: TObject);
    procedure VisualizarClick(Sender: TObject);
    procedure Button9Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

uses TransformaGrafo, GTSP, NegCircuit, CriaGrafo, LowerBound, Help,
  TransfSimples,AnimaGrafo, ConstRede;

{$R *.DFM}

Type
  TLink = Record
    From: Integer;
    Too:  Integer;
    Tipo: Integer;
    ArcContra: Integer;
    Cost: Integer;
    ModCost: Integer;
    ProxSuc: Integer;
    ProxPrd: Integer;
    NumUsed: Integer;
  End;
  TNode = Record
    FirstSuc: Integer;
    FirstPrd: Integer;
```

```
    Tipo: Integer;
    Grau: Integer;
    Linha: Integer;
    Coluna: Integer;
    Usado: Integer;
  End;
  TVet= Array[1..2000] Of Integer;

Var
   I,J,M,N,Q1,Q2,Q,NoI,NoJ,M1,K,K1,K2,MAPrevisto: Integer;
   NC,NL,MA,ME,I1,J1,PA,TotArc,TotLink,TipLink:Integer;
   OK:Boolean;
   Rede:TextFile;
   P,GM:Real;
   Vr:Variant;
   Arq,ArqNome,NomeAbr:String;
   Link: Array[1..10000] of TLink;
   Node: Array[1..2000] of TNode;
   Inf,Red,PAnR: Integer;
   CppCost,AddCost:Integer;
   ArqNomeTxt: String;


Function Rand(Num1,Num2:Integer):Integer;
Var R:Real;
Begin
   //R:=Num1+(Num2-Num1)*Random;
   //Rand:=Round(R);
   Rand:=Num1+Trunc((Num2-Num1+1)*Random);
End;


Procedure calculeGrau(Var I,I3,Gr:Integer);
Var GMG,R1,R2:Real;
Begin
   R1:=M-TotLink/2;
   R2:=N-I3+1;
   GMG:=R1/R2;
   If Node[I].Tipo=5 then GMG:=GMG*1.2;
   If Node[I].Tipo=3 then GMG:=2.0;
   GMG:=GMG*(1.5+Random)/2.;
   Gr:=Round(GMG);
   If (Node[I].Tipo=5) And (Gr>5) then Gr:=5;
End;


Procedure DefineLink(Var I,J:Integer);
Var IL,IC,JL,JC,DL,DC,K,T:Integer;
    Certo:Boolean;
Begin
   T:=0;
   Repeat
      T:=T+1;
      Certo:=true;
      IL:=Node[I].Linha;
      IC:=Node[I].Coluna;
      DL:=Rand(-1,1);
      DC:=Rand(-1,1);
      If IL=1 Then DL:=Rand(0,1);
      If IL=NL Then DL:=Rand(-1,0);
      If IC=1 Then DC:=Rand(0,1);
      If IC=NC Then DC:=Rand(-1,0);
      JL:=IL+DL;
      JC:=IC+DC;
      J:=(JL-1)*NC+JC;
      If J=I Then Certo:=false;
      If Node[J].Grau=Node[J].Tipo Then Certo:=false;
```

```
      If Node[J].Grau>GM+1 Then Certo:=false;
      K:=Node[J].FirstPrd;
      While K<>0 Do
        Begin
        If (Link[K].From=I) And (Link[K].Too=J)
          Then Certo:=false;
        K:=Link[K].ProxPrd;
        End;
      If T>50 Then Certo:=true;
    Until Certo;
    If T>20 Then J:=-J;
  End;


Procedure InicializaVet;
// Inicialização de vetores de encadeamento de nós e Links
Var I,J,I1:Integer;
Begin
    For I:=1 To NL Do
      For J:=1 To NC Do
        Begin
        I1:=(I-1)*NC+J;
        Node[I1].Linha:= I;
        Node[I1].Coluna:= J;
        Node[I1].firstSuc:= 0;
        Node[I1].firstPrd:= 0;
        Node[I1].Grau:= 0;
        Node[I1].Usado:= 0;
        Node[I1].Tipo:= 8;          {Tipo: No de canto=3, 'no de borda=5, no de meio=8}
        If (I-1)*(I-NL)*(J-1)*(J-NC)=0
          Then Node[I1].Tipo:= 5;
        If ((I-1)*(I-NL)=0) And ((J-1)*(J-NC)=0)
          Then Node[I1].Tipo:= 3;
        End;

// Inicialização dos vetores de encadeamento dos Links
    For J:=1 To M Do
      Begin
      I:=J*2-1;
      Link[I].From:= 0;
      Link[I].Too:= 0;
      Link[I].Tipo:= 0;
      Link[I].ArcContra:= I+1;
      Link[I].Cost:= Rand(Q1,Q2);
      Link[I].ModCost:= 0;
      Link[I].ProxSuc:= 0;
      Link[I].ProxPrd:= 0;
      Link[I].NumUsed:= 0;
      I:=I+1;
      Link[I].From:= 0;
      Link[I].Too:= 0;
      Link[I].Tipo:= 0;
      Link[I].ArcContra:= I-1;
      Link[I].Cost:= Link[I-1].Cost;
      Link[I].ModCost:= 0;
      Link[I].ProxSuc:= 0;
      Link[I].ProxPrd:= 0;
      Link[I].NumUsed:= 0;
      End;
End;


Procedure geraGrafo;
Var I,J,I1,J1,I2,I3,Gr:Integer;
    S,R:Real;
Begin
    N:=NC*NL;
```

```
  R:=N;
  M:=Round(R*(GM/2.0));    {Valor provisório para M}
  InicializaVet;
  Randomize;

  // Criar os Links *****************}
  For I2:=-NC+1 To -NC+N Do
    Begin
    I:=I2;
    I3:=I2+NC;
    If I<1 Then I:=I+N;     {Começa pela ultima linha do grafo, depois volta para a primeira a penultima}
    CalculeGrau(I,I3,Gr);
    If Gr>0 Then
     For I1:=1 To Gr Do
      Begin
      DefineLink(I,J);
      If (J>0) And (TotLink<2*M) Then
        Begin
        TotLink:=TotLink+1;
        K:=TotLink;    {Criando o arco K}
        Form1.StatusBar1.SimpleText := 'Gerando links numero '+IntToStr(K);
        Link[K].From:=I;
        Link[K].Too:=J;
        Link[K].ProxSuc:=Node[I].FirstSuc;
        Link[K].ProxPrd:=Node[J].FirstPrd;
        Node[I].FirstSuc:=K;
        Node[J].FirstPrd:=K;
        Node[I].Grau:=Node[I].Grau+1;
        TotLink:=TotLink+1;
        K:=TotLink;     {Criando o Arco Contra K}
        Link[K].From:=J;
        Link[K].Too:=I;
        Link[K].ProxSuc:=Node[J].FirstSuc;
        Link[K].ProxPrd:=Node[I].FirstPrd;
        Node[J].FirstSuc:=K;
        Node[I].FirstPrd:=K;
        Node[J].Grau:=Node[J].Grau+1;
        End;
      End;
    End;
  R:=TotLink;
  M:=Round(R/2);
  MA:=0;
  ME:=0;
  R:=PA*M;
  R:=R/100;
  MAPrevisto:=Round(R);
End;


Procedure DadosEntrada(Edit1,Edit2,EDit3,Edit4,
        Edit5,Edit6,Edit7,Edit8,Edit16: TEdit);
Begin
  NL := StrToInt(Edit1.Text);
  NC := StrToInt(Edit2.Text);
  Vr := Edit3.Text;
  GM := Vr;
  PA := StrToInt(Edit4.Text);
  PAnR := StrToInt(Edit16.Text);
  Q1 := StrToInt(Edit5.Text);
  Q2 := StrToInt(Edit6.Text);
  Red:= StrToInt(Edit7.Text);
  //Inf:= StrToInt(Edit8.Text);
  Inf:= 99999;
  TotArc:=0;
  TotLink:=0;
End;
```

```
Procedure GravaArq(ListBox1:TListBox;
          Label9,Label10,Label11: Tlabel;
          GroupBox2:TGroupBox);
Var Linha, Compl: String;
   ArcR, ArcNR, MAnR, Sorteio: Integer;

Begin
  {Gerar o Arquivo Contendo o Grafo *****************}
  If Copy(ArqNome,Length(ArqNome)-3,1)='.'
    Then ArqNomeTxt := ArqNome
    Else ArqNomeTxt := ArqNome+'.Txt';
  AssignFile(Rede,ArqNomeTxt);
  ReWrite(Rede);
  WriteLn(Rede,M,'  ',N,'  ',NL,'  ',NC,'  ',MA,'  ',ME,'  ',CppCost,'  ',AddCost);
  NomeAbr := ArqNome;
  If Length(NomeAbr) > 22
   Then Delete(NomeAbr,1,Length(NomeAbr)-22);
  NomeAbr := Copy(NomeAbr,1, Length(NomeAbr)-4);
  GroupBox2.Caption := 'Grafo ativo : ...'+NomeAbr;
  Label9.Caption :=  'Nós:      '+IntToStr(N);
  Label10.Caption := 'Arcos:    '+IntToStr(MA);
  Label11.Caption := 'Arestas: '+IntToStr(ME);
  GroupBox2.Update;
  ListBox1.enabled := True;
  ListBox1.Color := clWindow;
  ListBox1.Items.Clear;
  ListBox1.Items.Add ('Links='+IntToStr(M)+'  Nodes='+IntToStr(N));
  ListBox1.Items.Add ('  CppCost='+IntToStr(CppCost)+'  AddCost='+IntToStr(AddCost));
  MAnR:=Round((M*PAnR)/100.0);        //Num de arcos não-rqueridos a serem marcados
  If MAnR>MA then MAnR:=MA;
  ArcR:=0;                           //Num provisorio de arcos requeridos
  ArcNR:=0;                           // "      "          "   não  "
  For I:=1 To M Do
    Begin
    K:=2*I-1;
    If Link[K].Tipo=-1
      Then K:=Link[K].ArcContra;
    If (Link[K].Tipo=0) And (Link[K].Too<Link[K].From)
      Then K:=Link[K].ArcContra;
    With Link[K] Do
      Begin
      Write(Rede,I,'  ',Tipo,'  ',From,' ',Node[From].Linha,' ',Node[From].Coluna,'   ',
            Too,' ',Node[Too].Linha,' ',Node[Too].Coluna,'   ',Cost,'   ',NumUsed,'   ',Link[ArcContra].NumUsed);
      Compl:='';
      If Tipo=1 then
        begin         // sorteio para marcar o arco como não-requerido
        Sorteio := Rand(ArcNR, MA-ArcR-1);
        If Sorteio<MAnR
          then begin
              ArcNR:=ArcNR+1;
              Compl:='   $  0';
              end
          else ArcR:=ArcR+1;
        end;
      WriteLn(Rede,Compl);
      Linha :='';
      Linha := Linha + IntToStr(I) + '  ';
      Linha := Linha + IntToStr(Tipo) + '    ';
      Linha := Linha + IntToStr(From) + '  ';
      Linha := Linha + IntToStr(Too) + '  ';
      Linha := Linha + IntToStr(Cost) + '    ';
      Linha := Linha + IntToStr(NumUsed) + '  ';
      Linha := Linha + IntToStr(Link[ArcContra].NumUsed) + '  ';
      If Compl>'' Then Linha:= Linha + ' $ 0';
      ListBox1.Items.Add (Linha);
```

```
        End;
      End;
  CloseFile(Rede);
End;


Procedure EscolhaDoisNosNaoUsados(Var N1,N2,NNU:Integer; Var VetNU:TVet);
Var I:Integer;
Begin
  If NNU<0 Then
    Begin
    NNU:=0;
    For I:=1 To N Do
      Begin
      If Node[I].Tipo<8 Then
        Begin
        NNU:=NNU+1;
        VetNU[NNU]:=I;
        End;
      End;
    End;
  If NNU>1 Then
    Begin
    I:=Rand(1,NNU);
    N1:=VetNU[I];
    VetNU[I]:=VetNU[NNU];
    NNU:=NNU-1;
    I:=Rand(1,NNU);
    N2:=VetNU[I];
    VetNU[I]:=VetNU[NNU];
    NNU:=NNU-1;
    Node[N1].Usado:=1;
    Node[N2].Usado:=1;
    End;
End;


Procedure AchaCaminhoMinimo(Var S,T,CMin,CardCad:Integer; Var Cadeia:TVet);
Var I,J,P,X,K,Min,XStar,Z:Integer;
   Rotulo,SeqNo,SeqLink:Array[1..2000] Of Integer;
   Temp:Array[1..2000] Of Boolean;
Begin
  //Dijkstra's Algorithm, Christofides, pg. 152
  //Step 1.
  For I:= 1 To N Do
    Begin
    Rotulo[I]:=Inf;
    Temp[I]:=true;
    Cadeia[I]:=0;
    End;
  Rotulo[S]:=0;
  P:=S;
  SeqNo[P]:=0;
  SeqLink[P]:=0;
  While T<>P Do
    Begin
    //Step 2.
    K:=Node[P].FirstSuc;
    X:=Link[K].Too;
    While (K<>0) Do
      Begin
      If Temp[X] Then
        If Rotulo[X]>Rotulo[P]+Link[K].ModCost Then
          Begin
          Rotulo[X]:=Rotulo[P]+Link[K].ModCost;
          SeqNo[X]:=P;
          SeqLink[X]:=K;
```

```
            End;
        K:=Link[K].ProxSuc;
        X:=Link[K].Too;
        End;
      //Step 3.
      Min:=Inf;
      For I:=1 To N Do
        IF Temp[I]=true Then
          If Rotulo[I]<Min Then
            Begin
            Min:=Rotulo[I];
            XStar:=I;
            End;
      //Step 4.
      Temp[XStar]:=False;
      P:=XStar;
      //Step 5.
      End;
  CMin:=Rotulo[T];
  //Identificacao do Caminho
  J:=T;
  I:=N+1;
  While J<>S Do
    Begin
    I:=I-1;
    Cadeia[I]:=SeqLink[J];
    J:=SeqNo[J];
    End;
  CardCad:=N-I+1;
  For J:= 1 To CardCad Do
    Cadeia[J]:=Cadeia[J+I-1];
End;


Procedure AtualizaCustosCadeia(Var CardCad:Integer; Var Cadeia:TVet);
Var I,NumUsado,NumContra:Integer;
Begin
For I:=1 To CardCad Do
  With Link[Cadeia[I]] Do
    Begin
    NumUsado:=NumUsed;
    NumContra:=Link[ArcContra].NumUsed;
    If (NumUsado=0) And (NumContra=0) Then
      Begin
      If Rand(MA,M-ME-1)<MAPrevisto
        Then Begin
            MA:=MA+1;
            Tipo:=1;
            Link[ArcContra].Tipo:=-1;
            Link[ArcContra].ModCost:=Inf-1;
            End
        Else Begin
            ME:=ME+1;
            Link[ArcContra].ModCost:=Cost;
            End;
      ModCost:=Cost;
      NumUsed:=1;
      End;
    If (NumUsado>0) And (NumContra=0) Then
      Begin
      ModCost:=Cost;
      NumUsed:=NumUsed+1;
      Link[ArcContra].ModCost:=-Cost;
      End;
    If (NumUsado=0) And (NumContra=1) Then
      Begin
      ModCost:=-Cost;
```

```
        NumUsed:=NumUsed+1;
        Link[ArcContra].ModCost:=-Cost;
        End;
      If (NumUsado=0) And (NumContra>1) Then
        Begin
        Link[ArcContra].NumUsed:=Link[ArcContra].NumUsed-1;
        If (Link[ArcContra].NumUsed=1) Then
          Begin
          If Tipo=0 Then ModCost:=Cost;
          If Tipo=-1 Then ModCost:=Inf-1;
          End;
        End;
      If (NumUsado=1) And (NumContra=1) Then
        Begin
        ModCost:=Cost;
        Link[ArcContra].NumUsed:=0;
        Link[ArcContra].ModCost:=Cost;
        End;
      End;
End;


Procedure OrientaGrafo;
Var  I,N1,N2,NNU,CMin,CardCad:Integer;
     ExisteNoNaoUsado:Boolean;
     VetNU,Cadeia: TVet;
     Texto:String;
Begin
   NNU:=-1;
   ExisteNoNaoUsado:=true;
   While ExisteNoNaoUsado Do
      Begin
      EscolhaDoisNosNaoUsados(N1,N2,NNU,VetNU);
      Form1.StatusBar1.SimpleText := 'Orientando links a partir de nós não usados '+IntToStr(N1)+'  '+
                                          IntToStr(N2);
      If NNU<2 Then ExisteNoNaoUsado:=false;
      AchaCaminhoMinimo(N1,N2,CMin,CardCad,Cadeia);
      AtualizaCustosCadeia(CardCad,Cadeia);
      AchaCaminhoMinimo(N2,N1,CMin,CardCad,Cadeia);
      AtualizaCustosCadeia(CardCad,Cadeia);
      End;
   For J:= 1 To M Do
      Begin
      I:=2*J-1;
      If Link[I].NumUsed+Link[Link[I].ArcContra].NumUsed=0 Then
        Begin
        N1:=Link[I].From;
        N2:=Link[I].Too;
        Form1.StatusBar1.SimpleText := 'Orientando links a partir do linke não usado '+IntToStr(I);
        AchaCaminhoMinimo(N1,N2,CMin,CardCad,Cadeia);
        AtualizaCustosCadeia(CardCad,Cadeia);
        AchaCaminhoMinimo(N2,N1,CMin,CardCad,Cadeia);
        AtualizaCustosCadeia(CardCad,Cadeia);
        End;
      End;
   For J:= 1 To M Do
      Begin
      I:=2*J-1;
      If Link[I].NumUsed+Link[Link[I].ArcContra].NumUsed>1 Then
        Begin
        Form1.StatusBar1.SimpleText := 'Re-Orientando links a partir do linke usado '+IntToStr(I);
        N1:=Link[I].From;
        N2:=Link[I].Too;
        AchaCaminhoMinimo(N1,N2,CMin,CardCad,Cadeia);
        AtualizaCustosCadeia(CardCad,Cadeia);
        AchaCaminhoMinimo(N2,N1,CMin,CardCad,Cadeia);
        AtualizaCustosCadeia(CardCad,Cadeia);
```

```
        AchaCaminhoMinimo(N2,N1,CMin,CardCad,Cadeia);
        AtualizaCustosCadeia(CardCad,Cadeia);
        AchaCaminhoMinimo(N1,N2,CMin,CardCad,Cadeia);
        AtualizaCustosCadeia(CardCad,Cadeia);
        End;
      End;
End;


Procedure CalculaCusto;
Var I,J,Vezes:Integer;
Begin
  CppCost:=0;
  AddCost:=0;
  For J:=1 To M Do
    Begin
    I:=2*J-1;
    Vezes:=Link[I].NumUsed+Link[I+1].NumUsed;
    If Vezes>0 Then
      Begin
      CppCost:=CppCost+Vezes*Link[I].Cost;
      AddCost:=AddCost+(Vezes-1)*Link[I].Cost;
      End;
    End;
End;


Procedure ReduzCustoLinksDuplicados;
Var I,J,Vezes:Integer;
   V:Real;
Begin
  For J:=1 To M Do
    Begin
    I:=2*J-1;
    Vezes:=Link[I].NumUsed+Link[I+1].NumUsed;
    If Vezes>1 Then
      Begin
      V:=Link[I].Cost;
      V:=V*(100-Red);
      V:=V/100;
      Link[I].Cost:=Round(V);
      Link[I+1].Cost:=Round(V);
      End;
    End;
End;


Function Distancia(i,j:integer):Real;
begin
  Result:=Sqrt((Node[i].Linha-Node[j].Linha)*
            (Node[i].Linha-Node[j].Linha)+
            (Node[i].Coluna-Node[j].Coluna)*
            (Node[i].Coluna-Node[j].Coluna));
end;

Procedure VerifiqueConsistencia(Var GrafoNaoConsistente:Boolean);
Var I,J:Integer;
   Certo:Boolean;
Begin
  GrafoNaoConsistente:=true;
  Certo:=true;
  For I:=1 To 2*M Do
    If (Link[I].Tipo=-1) and (Link[I].NumUsed>0) Then Certo:=false;
  If Certo=true Then GrafoNaoConsistente:=false;
End;
```

```
Procedure CalculaCustoConversao(N1,N2,N3:Integer ; Var Penal:Integer);
var V1L,V1C,V2L,V2C,SenoTeta,CosTeta,A: Real;
   PenalFrente,PenalDireita,PenalEsquerda:Integer;
begin
  PenalFrente:=StrToInt(Form1.Edit13.Text);
  PenalDireita:=StrToInt(Form1.Edit14.Text);
  PenalEsquerda:=StrToInt(Form1.Edit15.Text);
  V1L:= Node[N2].Linha-Node[N1].Linha;
  V1C:= Node[N2].Coluna-Node[N1].Coluna;
  V2L:= Node[N3].Linha-Node[N2].Linha;
  V2C:= Node[N3].Coluna-Node[N2].Coluna;
  A:=Sqrt((V1L*V1L+V1C*V1C)*(V2L*V2L+V2C*V2C));
  SenoTeta:= (V1L*V2C-V1C*V2L)/A;
  CosTeta:=  (V1L*V2L+V1C*V2C)/A;
  If CosTeta>0.866 then penal:=PenalFrente;       // Em frente      (Angulo -30 a 30)
  If SenoTeta>=0.5 then penal:=PenalEsquerda;     // Para Esquerda  (Angulo 30 a 150)
  If CosTeta<-0.866 then penal:=PenalEsquerda;    // Esquerda / Direita com angulo fechado (Angulo 150 a
                                                  // 210)
  If SenoTeta<=-0.5 then penal:=PenalDireita;         // Para Direita   (Angulo 210 a 330)
end;


Procedure InserirConversoesViaveis(ListBox1:TListBox);
var N1,N2,N3,L,K,Penal: Integer;
   Linha: String;
Begin
  AssignFile(Rede,ArqNomeTxt);
  Append(Rede);
  WriteLn(Rede,'* Conversões Permitidas, ou Penalizadas');
  ListBox1.Items.Add ('Conversões Permitidas / Penalizadas');
  For I:=1 To N Do
    Begin
    K:=Node[I].FirstSuc;
    While K<>0 Do
      begin
      L:=0;
      If Link[K].Tipo<>-1 Then
        begin
        N1:=Link[K].From;
        N2:=Link[K].Too;
        L:=Node[N2].FirstSuc;
        end;
      While L<>0 Do
        Begin
        N3:=Link[L].Too;
        If N3=N1 Then N3:=0;
        If Link[L].Tipo=-1 Then N3:=0;   //N3 =0 -> Retorno U
        CalculaCustoConversao(N1,N2,N3,Penal);
        If N3>0 Then
          begin
          //WriteLn(Rede,N1,'  ',N2,'  ',N3,'       0');
          WriteLn(Rede,N1,'  ',N2,'  ',N3,'  ',Penal);
          Linha :='';
          Linha := Linha + IntToStr(N1) + '  ';
          Linha := Linha + IntToStr(N2) + '  ';
          Linha := Linha + IntToStr(N3) + '  ';
          Linha := Linha + IntToStr(Penal) + '  ';
          ListBox1.Items.Add (Linha);
          end;
        L:=Link[L].ProxSuc;
        end;
      K:=Link[K].ProxSuc;
      end;
    End;
  CloseFile(Rede);
end;
```

```
procedure TForm1.Button1Click(Sender: TObject);
Var I,J,Cont:Integer;
    GrafoNaoConsistente:Boolean;
    TempoInicio,TempoFinal:TDateTime;

begin
  For I:=1 To 2000 Do J:= Rand(0,99); {Inicialização do gerador de numeros aleatórios}
  Cont:=0;
  GrafoNaoConsistente:=true;
  While (Cont<10) And GrafoNaoConsistente Do
     Begin
     Cont:=Cont+1;
     DadosEntrada (Edit1,Edit2,EDit3,Edit4,Edit5,Edit6,Edit7,Edit8,Edit16);
     TempoInicio:=Time;
     GeraGrafo;
     OrientaGrafo;
     VerifiqueConsistencia(GrafoNaoConsistente);
     If GrafoNaoConsistente Then
        ShowMessage('Falha na tentativa '+IntToStr(Cont)+'/10 de construir o grafo');
     End;
  ReduzCustoLinksDuplicados;
  CalculaCusto;
  TempoFinal:= Time;
  Label15.Caption:='Tempo CPU:    '+TimeToStr(TempoFinal-TempoInicio);
  StatusBar1.SimpleText := 'Escolhe um nome para o arquivo a ser gravado';
//  saveDialog1.InitialDir := 'C:\Arquivos de programas\Borland\Delphi4\Projects\DadosMcpp\';
  If saveDialog1.Execute
    Then ArqNome := saveDialog1.FileName
    Else Exit;
  GravaArq(ListBox1,Label9,Label10,Label11,GroupBox2);
  If CheckBox2.Checked = true then InserirConversoesViaveis(ListBox1);
  If CheckBox4.Checked = true then form10.Button1.Click;
  MessageBeep($FFFFFFFF);
  StatusBar1.SimpleText := 'O Grafo foi gerado com sucesso';
  RadioButton2.Checked := true;
  Radiogroup1.OnClick := RadioGroup1Click;
end;


procedure TForm1.Button2Click(Sender: TObject);
begin
  form2.Button1.Click;
  RadioButton3.Checked := true;
  Radiogroup1.OnClick := RadioGroup1Click;
end;


procedure TForm1.Button8Click(Sender: TObject);
begin
  form9.Button1.Click;
  RadioButton3.Checked := true;
  Radiogroup1.OnClick := RadioGroup1Click;
end;


procedure TForm1.Button3Click(Sender: TObject);
begin
  //  form3.show;
  form3.Button1.Click;
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
  //  form3.show;
  form4.Button1.Click;
```

```
end;


procedure TForm1.Button5Click(Sender: TObject);
begin
  form5.Button1.Click;
end;


procedure TForm1.Button6Click(Sender: TObject);
begin
  form6.Button1.Click;
end;


procedure TForm1.RadioGroup1Click(Sender: TObject);
begin
  GroupBox1.enabled := false;
  GroupBox2.enabled := true;
  GroupBox4.enabled := false;
  If RadioButton1.Checked Then
    Begin
    Button1.Enabled := true;
    Button5.Enabled := true;
    Button2.Enabled := false;
    Button3.Enabled := false;
    Button4.Enabled := false;
    Button6.Enabled := false;
    Button7.Enabled := false;
    Button8.Enabled := false;
    Button9.Enabled := true;
    GroupBox1.enabled := true;
    GroupBox4.enabled := false;
    Edit1.Enabled := True;
    Edit2.Enabled := True;
    Edit3.Enabled := True;
    Edit4.Enabled := True;
    Edit5.Enabled := True;
    Edit6.Enabled := True;
    Edit7.Enabled := True;
    End;
  If RadioButton2.Checked Then
    Begin
    Button1.Enabled := false;
    Button2.Enabled := true;
    Button6.Enabled := true;
    Button3.Enabled := false;
    Button4.Enabled := false;
    Button5.Enabled := false;
    Button7.Enabled := false;
    Button8.Enabled := true;
    Button9.Enabled := false;
    GroupBox1.enabled := false;
    GroupBox4.enabled := true;
    End;
  If RadioButton3.Checked Then
    Begin
    Button1.Enabled := false;
    Button2.Enabled := false;
    Button3.Enabled := true;
    Button4.Enabled := true;
    Button5.Enabled := false;
    Button6.Enabled := false;
    Button7.Enabled := true;
    Button8.Enabled := false;
    Button9.Enabled := false;
    GroupBox1.enabled := false;
```

```
    GroupBox4.enabled := false;
    End;
end;

procedure TForm1.Button7Click(Sender: TObject);
begin
  FrmPrincipal.Show;
end;

procedure TForm1.BitBtn1Click(Sender: TObject);
begin
  Form7.Show;
  Form7.ListBox1.Visible:=true;
end;

procedure TForm1.VisualizarClick(Sender: TObject);
begin
  form10.Button4.Click;
end;

procedure TForm1.Button9Click(Sender: TObject);
begin
  //Form11.Show;
  //form11.Button1.Click;
  Form11.Visible:=true;
end;

End.
```

```
// Programa para transformar um grafo misto num grafo completo
// no formato exigido pelos programas GTSP e UCityList (SCR9)
// Este módulo permite a inserção de restrições nos vértices
// e especificação dos nós de inicio e fim do roteiro.
// Se o nó inicio da rota for diferente do nó final, um arco
// artificial na forma de (FimRota, InicioRota) é acrescido,
// antes de efetuar a transformação.
// A transformação pode ser efetuada para o Problema do carteiro
// Rural- RPP. Se o problema é RPP, automaticamente é identificado
// e acionada a rotina ReduzirRuralPostman que reduz a matriz final
// para os link requeridos.
// Versão 4, Agosto de 2003.

unit TransformaGrafo;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  Tform2 = class(TForm)
    Button1: TButton;
    OpenDialog1: TOpenDialog;
    SaveDialog1: TSaveDialog;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  form2: Tform2;

implementation

uses URandGraf,TransfSimples;

{$R *.DFM}

Type
   TArc = Record
     Noi:   Integer;
     Nof:   Integer;
   end;
   TLink = Record
     From:       Integer;      // Nó inicial do Link
     Too:        Integer;      // Nó final do Link
     Custo:      Integer;      // Custo Original do Link
     Arc:        TArc;         // Arco transformado associado, com novos nós terminais
     ArcContra:  Integer;      // Correspondencia com Arco no sentido opsto
                               // Link[i].ArcContra = k, então aresta i é representada pelo
                               // par de arcos i , k.  Se =-1, então i originalmente é arco
     Tipo:       Integer;      // 1:Arco,  0:Aresta,  2: Aresta sentido oposto
     Req:        Integer;      // 1:Link Requerido    0: Link não requerido
     ArcContraReq: Integer;    // É a mesma ArcContra, calculada na base de apenas Links
                               // requeridos. para os não requeridos seu valor é zero.
   end;

Var
   Link: Array[1..4000] of TLink;
   UmLink: TLink;

   L:array[1..4000,1..4000] of Integer;
   I,J,K,I1,J1,d,Noi,Nof,N,M,M1,M2,NL,NC,MA,ME,a,b: Integer;
```

```
  Rede,RedeOrig,RedeTemp:TextFile;
  Arq,ArqNome,NomeAbr,ArqAux,ArqNomeTxt,ArqTemp:String;
  Inf,MenosInf,Penalidade:Integer;
  InicioRota, FimRota: Integer;
  ArqErrado,Rural:Boolean;
Const
  NumMaxNos=4000;


Procedure ReduzirRuralPostman;
// Esta rotina é chamada se o grafo possuir pelo menos um link
// não requerido. Ele reduz a matriz final, de acordo com os links
// requeridos e atualiza os valores de M1, MA e ME (links, arcos e
// arestas requeridos + arco artificial)
var a,I,c,b,J,II,JJ:Integer;
  NoFinalAresta:  Integer;
  MAR,MER,M1R,M2R:Integer;
  OrdemLinkR:     Integer;
  Cont:           Array[1..2000] of integer;
Begin
  // M1R: Número de links requeridos do grafo original M1R= MAR+MER
  // MAR: Número de Arcos    "       "        "
  // MER: Numero de Arestas   "       "        "
  MAR:=0;
  MER:=0;
  For I:=1 To M1 Do
    If Link[I].Req=1 then
      begin
      If Link[I].Tipo=1 then MAR:=MAR+1;
      If Link[I].Tipo=0 then MER:=MER+1;
      end;
  M1R:=MAR+MER;
  M2R:=M1R+MER;
  OrdemLinkR:=0;
  NoFinalAresta:=M1R;
  For a:=1 To M1 Do                  // Criando a nova correspondencia entre
    Begin                          // o link a e seu respectivo
    If Link[a].Req=1                // Link[a].ArcContra, armazenando no
     then OrdemLinkR:=OrdemLinkR+1;  // vetor Link[a].ArcContraReq
    If Link[a].Tipo=1
      Then begin
          If Link[a].Req=1
            Then Link[a].ArcContraReq:=-1
            Else Link[a].ArcContraReq:=0;
          end
      Else begin
          c:=Link[a].ArcContra;
          If Link[a].Req=1
            Then begin
                NoFinalAresta:=NoFinalAresta+1;
                Link[a].ArcContraReq:=NoFinalAresta;
                Link[c].ArcContraReq:= OrdemLinkR;
                end
            else begin
                Link[a].ArcContraReq:=0;
                Link[c].ArcContraReq:=0;
                end;
          end;
    End;
//   For I:=1 To M2 Do
//     Cont[I]:= Link[I].ArcContraReq;

  //Eliminar linhas/colunas não requeridas da matriz L
  II:=0;
  For I:=1 to M2 Do
    begin
    If Link[I].ArcContraReq <> 0 Then
```

```
        begin
        II:=II+1;
        JJ:=0;
        For J:=1 to M2 do
          begin
          If Link[J].ArcContraReq <> 0 Then
            begin
            JJ:=JJ+1;
            Cont[JJ]:=L[I,J];
            end;
          end;
        For J:=1 to JJ do
          L[II,J]:=Cont[J];
        Link[II].ArcContra:=Link[I].ArcContraReq
        end;
      end;

  M1:=M1R;
  M2:=M2R;
  MA:=MAR;
  ME:=MER;
End;


Procedure LerArq;
// Este procedimento lê o arquivo temporário ArqTemp, gerado na rotina
// DefineInicioEFimRoteiro. Este arquivo é identico ao arquivo original
// exceto um arco artificial que tem a mais (FimRota, InicioRota).
// A transformação é feita sobre o ArqTemp.
var Tip,I,J,I1,J1,II:Integer;
  NoFinalAresta: Integer;
  NoFinalArco:   Integer;
  NoAux:         Integer;
  NoInicioLink:  Integer;
  MA1,ME1:       Integer;
  Simbolo:       String[1];
  Requerido:     Integer;
Begin
  {Leitura de Arquivo / Inicializa‡Æo de Vetores **************}
  // M1: Número de links do grafo original M1= MA+ME
  // MA: Número de Arcos    "       "
  // ME: Numero de Arestas   "       "
  // M2: Número de nós (dimensão da matriz) na forma final transformada
  //     M2= M1+ME
  AssignFile(Rede,ArqTemp);
  Reset(Rede);
  Read(Rede,M1,N,NL,NC,MA,ME);
  Repeat
    Read(Rede,Simbolo);
  Until Simbolo='#';
  ReadLn(Rede,InicioRota,FimRota);
  If 2*(M1+ME)>NumMaxNos Then
    Begin
    Application.messagebox('Dimensão da Matriz Ultrapassa a Estabelecida no Programa.','Atualize o
                                    Programa',MB_OK);
    Halt;
    End;
  M2:=M1+ME;
  For I:=1 To 2*M2 Do
    Begin
    Link[I].ArcContra:=-Inf;
    For J:=1 To 2*M2 Do
      L[I,J]:=Inf;
    End;
  NoFinalAresta:=M1;
  NoFinalArco:=M1+ME;
  NoAux:=2*M1;
```

```
  MA1:=0;
  ME1:=0;
  For II:=1 To M1 Do
    Begin
    Readln(Rede,a,Tip,Noi,I,J,Nof,I1,J1,d,Requerido);
    NoInicioLink:=a;
    Link[a].From:=Noi;
    Link[a].Too:=Nof;
    Link[a].Custo:=d;
    Link[a].Req:=Requerido;
    If Tip=1
    Then begin
          Link[a].Tipo:=1;
          Link[a].Arc.Noi:=NoInicioLink;
          NoFinalArco:=NoFinalArco+1;
          Link[a].Arc.Nof:= NoFinalArco;
          Link[a].ArcContra:=-1;
          L[Link[a].Arc.Noi, Link[a].Arc.Nof] := d;
          L[Link[a].Arc.Noi, Link[a].Arc.Noi] := 0;
          MA1:=MA1+1;
        end
    Else begin
          Link[a].Tipo:=0;
          Link[a].Arc.Noi:=NoInicioLink;
          NoFinalAresta:=NoFinalAresta+1;
          Link[a].Arc.Nof:= NoFinalAresta;
          b:=NoFinalAresta;
          Link[a].ArcContra:=b;

          Link[b].Tipo:=2;
          Link[b].From:=Nof;
          Link[b].Too:=Noi;
          Link[b].Custo:=d;
          Link[b].Req:=Requerido;
          Link[b].Arc.Noi:= NoAux+1;
          Link[b].Arc.Nof:= NoAux+2;
          NoAux:=NoAux+2;
          Link[b].ArcContra:=a;
          ME1:=ME1+1;

          L[Link[b].Arc.Noi, Link[b].Arc.Nof] := d;
          L[Link[b].Arc.Noi, Link[b].Arc.Noi] := 0;
          L[Link[a].Arc.Noi, Link[a].Arc.Nof] := d;
          L[Link[a].Arc.Noi, Link[a].Arc.Noi] := 0;
        end;
    End;
  CloseFile(Rede);
  If (MA1<>MA) Or (ME1<>ME) Then
    begin
    Application.messagebox('Número de arcos ou arestas incorreto','Programa será fechado!',MB_OK);
    Halt;
    end;
End;


Procedure DefineInicioEFimDoRoteiro(Edit11,Edit12:TEdit);
// Este procedimento lê os nós especificados para o início e fim da rota
// A partir disso cria um novo arco artificial (FimRota, InicioRota)
// com custo igual ao valor usado como infinito (-MenosInf).
// O arco artificial é criado, mesmo se FimRota = InicioRota.
// Gera um novo arquivo ArqTemp que é igual ao arquivo original,
// só que com um arco a mais (M1+1 links e MA+1 arcos)
// No final da primeira linha de ArqTemp, depois dos dados gerais do grafo,
// aparece o símbolo # e depois os nós inicial e final da rota.
Var Caractere: Char;
   II,NoRepetidoi,NoRepetidoj: Integer;
   InicioRotaCoordI,InicioRotaCoordJ,
```

```
    FimRotaCoordI,FimRotaCoordJ,Requerido: Integer;
    a,Tip,Noi,I,J,Nof,I1,J1,d,JJ: Integer;
    N1,N2,N3,Custo: Integer;
    St1:String[1];
begin
   InicioRota:=StrToInt(Form1.Edit11.Text);
   FimRota:=StrToInt(Form1.Edit12.Text);
   Assign(Rede,ArqNome);
   Reset(Rede);
   ArqTemp:='ArqTemp.Txt';
   AssignFile(RedeTemp,ArqTemp);
   ReWrite(RedeTemp);
   ReadLn(Rede,M1,N,NL,NC,MA,ME);
   WriteLn(RedeTemp,M1+1,'  ', N,'  ',NL,'  ',NC,'  ',MA+1,'  '
                ,ME,'  ','  # ',InicioRota,'  ',FimRota);
   InicioRotaCoordI:=-1;
   InicioRotaCoordJ:=-1;
   FimRotaCoordI:=-1;
   FimRotaCoordJ:=-1;

   Rural:=false;
   For II:=1 To M1 Do
      Begin
      Requerido:=1;
      Read(Rede,a,Tip,Noi,I,J,Nof,I1,J1,d);
      St1:=' ';
      JJ:=0;
      Repeat
         JJ:=JJ+1;
         Read(Rede,St1);
      Until (St1='$') or (JJ>50);
      If St1 = '$' then Read(Rede,Requerido);
      If Requerido=0 Then Rural:=true;
      ReadLn(Rede);
      N1:=Noi;
      Writeln(RedeTemp,a,'  ',Tip,'  ',Noi,' ',I,' ',J,'  '
                   ,Nof,' ',I1,' ',J1,'  ',d, '  ',Requerido);

      N1:=Noi;   // Identificando as coordenadas do InicioRota, FimRota
      If Noi=InicioRota then
         begin
         InicioRotaCoordI:=I;
         InicioRotaCoordJ:=J;
         end;
      If Nof=FimRota then
         begin
         FimRotaCoordI:=I1;
         FimRotaCoordJ:=J1;
         end;
      If (Tip=0) and (Noi=FimRota) then
         begin
         FimRotaCoordI:=I;
         FimRotaCoordJ:=J;
         end;
      If (Tip=0) and (Nof=InicioRota) then
         begin
         InicioRotaCoordI:=I1;
         InicioRotaCoordJ:=J1;
         end;
      End;
   If (InicioRotaCoordI=-1) Or (FimRotaCoordI=-1) Then
      begin
      Application.messagebox('Inicio ou Final especificado para a rota inválido','Programa será
fechado!',MB_OK);
      Halt;
      end;
   Writeln(RedeTemp,a+1,'  1  ',FimRota,' ',FimRotaCoordI,' ',FimRotaCoordJ,'  '
```

```
                ,InicioRota,' ',InicioRotaCoordI,' ',InicioRotaCoordJ,'  '
                ,-MenosInf, '  1');

  Caractere:=' ';
  If Not EoF(Rede) Then
    ReadLn(Rede,Caractere);
  If Caractere<>'*'
   Then
    begin
    CloseFile(Rede);
    CloseFile(RedeTemp);
    Exit;
    end
   Else
    WriteLn(RedeTemp,'* Conversões Permitidas, ou Penalizadas');
  While Not EoF(Rede) Do
    Begin       // Inserindo as conversoes permitidas, com respeito ao arco artificial
    NoRepetidoi:=0;
    NoRepetidoj:=0;
    ReadLn(Rede, N1,N2,N3,Custo);
    WriteLn(RedeTemp,N1,'  ',N2,'  ',N3,'  ',Custo);
    If (N3=FimRota) and (N2<>NoRepetidoi) Then
      WriteLn(RedeTemp,N2,'  ',N3,'  ',InicioRota,'  0');
    NoRepetidoi:=N2;
    If (N1=InicioRota) and (N2<>NoRepetidoj) Then
      WriteLn(RedeTemp,FimRota,'  ',N1,'  ',N2,'  0');
    NoRepetidoj:=N2;
    end;
  Close(Rede);
  CloseFile(RedeTemp);
  end;


Function CustoConversao(No1, No2, No3: Integer):Integer;
var caracter:String[1];
   NaoAchou:Boolean;
   N1,N2,N3,Custo:Integer;
begin
  custo:=0;
  ArqErrado:=false;
  If Form1.CheckBox3.Checked=false Then Exit;
  //Assign(Rede,ArqNome);
  Assign(Rede,ArqTemp);
  Reset(Rede);
  Caracter:='';
  While (Caracter<>'*') And (Not EoF(Rede)) Do
    ReadLn(Rede,Caracter);
  If Caracter<>'*' Then
    begin
    Application.messagebox('Verifique se o arquivo contém a lista das conversões viáveis'
                  ,'Formato do Arquivo Incorreto',MB_OK);
    Close(Rede);
    ArqErrado:=true;
    Exit;
    end;
  NaoAchou:=true;
  While (Not EoF(Rede)) And (NaoAchou) Do
    Begin
    ReadLn(Rede, N1,N2,N3,Custo);
    If N1=No1 Then
      If N2=No2 Then
        If N3=No3 Then
          NaoAchou:=false;
    end;
  If NaoAchou Then Custo:=Penalidade;
  Result:=Custo;
  Close(Rede);
```

```
end;


Procedure Transform;
var Pro:Boolean;
Begin
  M2:=M1+ME;
  //Cria‡Æo de Arcos de ConexÆo  ***********************
  For a:=1 To M2 Do
    Begin
    For b:=1 To M2 Do
     If Link[a].Too = Link[b].From Then
         begin
         L[Link[a].Arc.Nof, Link[b].Arc.Noi] := 0;
         // Checando para ver se o arco de conexão é de um retorno U
         If (Link[a].From = Link[b].Too) And (Form1.CheckBox1.Checked)
           Then L[Link[a].Arc.Nof, Link[b].Arc.Noi] := Penalidade;
         // Checando para ver se o arco de conexão é uma Conversão diferente
         If (Link[a].From <> Link[b].Too) And (Form1.CheckBox3.Checked)
           Then L[Link[a].Arc.Nof, Link[b].Arc.Noi] :=
               CustoConversao(Link[a].From, Link[a].Too, Link[b].Too);
         If ArqErrado Then Exit
         end;
    End;
End;


Procedure Floyd;
Begin
  Form1.ProgressBar1.Enabled:=true;
  Form1.ProgressBar1.Position:=0;
  For K:=1 To 2*M2 Do
    Begin
    For I:=1 To 2*M2 Do
      If (I<>K) And (L[I,K]<>Inf) Then
        For J:=1 To 2*M2 Do
          If (J<>K) And (L[K,J]<>Inf) Then
            If L[I,K]+L[K,J]<L[I,J] Then L[I,J]:=L[I,K]+L[K,J];
    Form1.ProgressBar1.StepBy(10000 div M2);
    End;
End;


Procedure AjusteGTSP;
Begin
  For I:=1 To M2 Do
    Begin
    // unificando os nós inicial e final de arcos (originalmente arcos)
    If Link[i].Tipo=1
      Then Begin
          J:=Link[I].Arc.Nof;
          For K:=1 To 2*M2 Do
            L[I,K]:=L[J,K];
          End;
    If Link[i].Tipo=0
    // unificando o nó inicial de arco i ao nó final do arco contra (originalmente aresta)
      Then Begin
          J:=Link[Link[I].ArcContra].Arc.Nof;
          For K:=1 To 2*M2 Do
            L[I,K]:=L[J,K];
          End;
    If Link[i].Tipo=2
    // unificando o nó inicial de arco contra i ao nó final do arco (originalmente aresta)
      Then Begin
          J:=Link[I].Arc.Noi;
          For K:=1 To 2*M2 Do
            L[K,I]:=L[K,J];
```

```
            End;
      End;
    For I:= 1 To M1 Do
      Begin
      If Link[I].Tipo=0 Then
        Begin
        J:=Link[I].Arc.Noi;
        K:=Link[I].Arc.Nof;
        L[J,K]:=MenosInf;
        L[K,J]:=MenosInf;
        b:=Link[I].ArcContra;
        Link[b].arc.Noi:=K;
        Link[b].arc.Nof:=J;
        End;
      End;
End;


Procedure GravaArq(SaveDialog1: TSaveDialog;
                   OpenDialog1:TOpenDialog;
                   ListBox1:TlistBox);
Var Frase,TipoArq:String;
    ValorFixo:Integer;
    II,JJ,M1Orig: Integer;
    a,Tip,Noi,I,J,Nof,I1,J1,d,Requerido: Integer;
    Linha:String[100];
Begin
  ValorFixo:=StrToInt(Form1.Edit9.Text);

  ArqNome:=OpenDialog1.FileName;
  TipoArq:='N';
  If Form1.CheckBox1.Checked Then TipoArq:='U';
  If Form1.CheckBox3.Checked Then
    If TipoArq='N'
      Then TipoArq:='C'
      Else TipoArq:='X';
  Insert(TipoArq,ArqNome,Length(ArqNome)-3);
  saveDialog1.FileName := ArqNome;
  If saveDialog1.Execute
    Then ArqNome := saveDialog1.FileName
    Else Exit;
  If Copy(ArqNome,Length(ArqNome)-3,1)='.'
    Then ArqNomeTxt := ArqNome
    Else ArqNomeTxt := ArqNome+'.Txt';
  AssignFile(Rede,ArqNomeTxt);
  ReWrite(Rede);
  WriteLn(Rede,M1,'  ',M2,'  ',NL,'  ',NC,'  ',MA,'  ',ME,'  ',ValorFixo,'  ',MenosInf);

  ListBox1.enabled := True;
  ListBox1.Color := clWindow;
  ListBox1.Items.Clear;
  If M2>20 Then Form1.ListBox1.Items.Add('Arquivo Grande. Visualize com WordPad');
  For I:=1 To M2 Do
    Begin
    Write(Rede,Link[I].ArcContra:5,'   ');
    Frase:='';
    For J:=1 To M2 Do
      begin
      Write(Rede,L[I,J]+ValorFixo:5,' ');
      If M2<21 Then Frase:=Frase+IntToStr(L[I,J])+'  ';
      end;
    WriteLn(Rede);
    If M2<21 Then Form1.ListBox1.Items.Add(Frase);
    End;

  // Grava o arquivo original, considerando apenas os links requeridos
  // na mesma ordem que aparecem (exclui os não requeridos) e acrescenta
```

```
  // o arco artificial no final da lista
  WriteLn(Rede,'$ Arquivo Original (Links Requeridos)');
  AssignFile(RedeOrig,ArqTemp);
  Reset(RedeOrig);
  II:=0;
  Read(RedeOrig,M1Orig,N,NL,NC,i,i);
  Write(Rede,M1,'  ',N,'  ',NL,'  ',NC,'  ',MA,'  ',ME,'  ');
  ReadLn(RedeOrig,Linha);
  WriteLn(Rede,Linha);
  For JJ:= 1 To M1Orig do
    begin
    Readln(RedeOrig,a,Tip,Noi,I,J,Nof,I1,J1,d,Requerido);
    If Requerido = 1 then
      begin
      II:=II+1;
      Writeln(Rede,II,'  ',Tip,'  ',Noi,'  ',I,'  ',J,'  ',Nof,'  '
            ,I1,'  ',J1,'  ',d);
      end;
    end;
  CloseFile(Rede);
  CloseFile(RedeOrig);
End;


procedure Tform2.Button1Click(Sender: TObject);
Var NomeAbr:String;
   TempoInicio,TempoFinal:TDateTime;
begin
  Inf:=99999;
  MenosInf:=-StrToInt(Form1.Edit8.Text);
  Penalidade:=StrToInt(Form1.Edit10.Text);
  Form1.StatusBar1.SimpleText := 'Escolhe o Grafo a ser lido';
  Form1.ListBox1.Enabled :=true;
  Form1.ListBox1.Items.Clear;
  //OpenDialog1.InitialDir := 'C:\Arquivos de programas\Borland\Delphi4\Projects\DadosMcpp\';
  OpenDialog1.InitialDir := 'C:\Meus documentos\Doutorado\MCPPgls Demo\';
  If OpenDialog1.Execute
    Then ArqNome := OpenDialog1.FileName
    Else Exit;
  Form1.StatusBar1.SimpleText := 'Transformando o grafo';
  DefineInicioEFimDoRoteiro(Form1.Edit11,Form1.Edit12);
  LerArq;
  Form1.GroupBox2.Enabled:=true;
  NomeAbr:=ArqNome;
  If Length(ArqNome)>20
     Then NomeAbr := Copy(ArqNome,Length(ArqNome)-19, 16);
  Form1.GroupBox2.Caption := 'Grafo ativo : ...'+NomeAbr;
  Form1.Label9.Caption :=  'Nós:      '+IntToStr(N);
  Form1.Label10.Caption := 'Arcos:    '+IntToStr(MA);
  Form1.Label11.Caption := 'Arestas: '+IntToStr(ME);
  Form1.GroupBox2.Update;
  TempoInicio:=Time;
  ArqErrado:=false;
  Transform;
  If ArqErrado Then exit;
  Form1.StatusBar1.SimpleText := 'O Grafo já foi transformado. Calculando agora a matriz de distâncias';
  Floyd;
  AjusteGTSP;
  TempoFinal:= Time;
  Form1.Label15.Caption:='Tempo CPU:    '+TimeToStr(TempoFinal-TempoInicio);
  Form1.StatusBar1.SimpleText := 'Salve o grafo transformado com um nome diferente';
  If Rural then ReduzirRuralPostman;
  GravaArq(SaveDialog1,OpenDialog1,Form1.ListBox1);

  NomeAbr := ArqNomeTxt;
  If Length(NomeAbr) > 20
    Then Delete(NomeAbr,1,Length(NomeAbr)-20);
```

```
//  NomeAbr := Copy(NomeAbr,1, Length(NomeAbr)-4);
  Form1.GroupBox2.Caption := 'Grafo ativo : ...'+NomeAbr;
  Form1.Label9.Caption :=  'Nós:      '+IntToStr(M2);
  Form1.Label10.Caption := 'Arcos:    '+IntToStr(M2*M2);
  Form1.Label11.Caption := 'Arestas: '+IntToStr(0);
  Form1.GroupBox2.Update;
  Form1.StatusBar1.SimpleText := 'O Grafo já foi gravado com sucesso';
  Form1.ProgressBar1.Position:=0;
end;

end.
```

```
// Esta Unidade contém as rotinas de representação gráfica
// dos grafos manhattan, e a animação dos roteiros.
// Versão 2. Junho de 2003.

unit AnimaGrafo;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls, Math, ExtDlgs;

type
  TForm10 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    OpenDialog1: TOpenDialog;
    OpenPictureDialog1: TOpenPictureDialog;
    Image1: TImage;
    PaintBox1: TPaintBox;
    Button5: TButton;
    procedure Button1Click(Sender: TObject);
    procedure PaintBox1Paint(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form10: TForm10;

implementation

uses Urandgraf, NegCircuit, UPrincipal, ConstRede;

{$R *.DFM}
Type
  TLink = Record
    Tipo: Integer;         // Tipo de Link no arquivo original:
                       // 0: aresta   1: arco
    Noi:  Integer;
    Nof:  Integer;
  End;
  TNo = Record
    ci:Integer;           // coordenada i
    cj:Integer;           // coordenada j
  End;
  TRota = Record
    Noi:  Integer;
    Nof:  Integer;
    Tipo: String[1];       // Tipo de link no arquivo rota: + - > D
    Arc:  Integer;
    Ordem:Integer;
    FreqTotal:Integer;
  end;

Var Raio:Integer;
    Rede,Roteiro:TextFile;
    ArqNome,ArqSol,ArqMapa:String;
    CompArco,PenaNormal,PenaDiag: Integer;
```

```
   M,N,NL,NC,MA,ME:Integer;
   BotaoAcionado:Integer;
   Velocidade,VelInversa: Integer;
   Falha: Boolean;

Const MaxLink = 2000;
      MaxRota = 2400;
      MaxNo   = 1000;


Function Delay(N:Integer):Real;
Var x:Real;
   i,j:Integer;
Begin
  X:=1.0;
  For i:=1 to 1*N do
    For j:=1 to 100+CompArco do
      x:=x+0.000001;
  Result:=x;
End;


Procedure CriaNo(x,y:Integer;PaintBox1:TPaintBox);
// Desenha um nó do grafo, com as coordenadas dadas
begin
  PaintBox1.Canvas.Pen.Width:=PenaDiag;
  PaintBox1.Canvas.Pen.Color:=clMaroon    ;
  PaintBox1.Canvas.Ellipse(x-Raio,y-Raio,x+Raio,y+Raio);
end;


Procedure CriaAresta(x1,y1,x2,y2:Integer;PaintBox1:TPaintBox);
// Desenha uma aresta original do grafo, com as coordenadas dadas
begin
  PaintBox1.Canvas.Pen.Width:=PenaDiag;
  //PaintBox1.Canvas.Pen.Color:=8400000;
  PaintBox1.Canvas.Pen.Color:=40000;
  If (x1 = x2) or (y1 = y2)
     Then PaintBox1.Canvas.Pen.Width:=PenaNormal;
  PaintBox1.Canvas.MoveTo(x1,y1);
  PaintBox1.Canvas.LineTo(x2,y2);
end;


Procedure CriaArco(x1,y1,x2,y2:Integer;PaintBox1:TPaintBox);
// Desenha um arco original do grafo, com as coordenadas dadas
var i,xs,ys,dx,dy,LargSeta: Integer;
   ponto1x,ponto2x,ponto3x,ponto1y,ponto2y,ponto3y:Integer;
begin
  PaintBox1.Canvas.Pen.Width:=PenaDiag;
  PaintBox1.Canvas.Pen.Color:=40000;
  //PaintBox1.Canvas.pen.Style := psClean;       Limpa a linha existente
  If (x1 = x2) or (y1 = y2)
     Then PaintBox1.Canvas.Pen.Width:=PenaNormal;
  PaintBox1.Canvas.MoveTo(x1,y1);
  PaintBox1.Canvas.LineTo(x2,y2);
  xs:=Round((x2-x1)*0.85)+x1;
  dx:=Round((x2-x1)/Abs(x2-x1+0.1));
  dy:=Round((y2-y1)/Abs(y2-y1+0.1));
  ys:=Round((y2-y1)*0.85)+y1;
  LargSeta:=Round(0.02*Sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1)+0.0));
  If dx*dy=0 Then
    begin
    xs:=Round((x2-x1)*0.80)+x1;
    ys:=Round((y2-y1)*0.80)+y1;
    LargSeta:=Round(2*LargSeta);
    end;
```

```
  Ponto1x:=xs-dy*LargSeta;
  Ponto1y:=ys+dx*LargSeta;
  Ponto2x:=xs+dy*LargSeta;
  Ponto2y:=ys-dx*LargSeta;
  Ponto3x:=Round((x2-x1)*0.93)+x1;
  Ponto3y:=Round((y2-y1)*0.93)+y1;
  If dx*dy=0 Then
    begin
    Ponto3x:=Round((x2-x1)*0.90)+x1;
    Ponto3y:=Round((y2-y1)*0.90)+y1;
    end;
  PaintBox1.Canvas.Polygon([point(ponto1x,ponto1y),point(ponto2x,ponto2y),
     point(ponto3x,ponto3y)]);
end;


Procedure CalculaCompArco;
// Calcula comprimento de cada link, em função do tamanho do grafo
begin
   PenaNormal:=3;                // Espessura da caneta
   PenaDiag:=2;
   If BotaoAcionado=0 then
      begin
      CompArco:=1;
      Exit;
      end;
   CompArco:=min((Screen.Width div (NC+1)), (Screen.Height div (NL+1)));
   Raio:=CompArco div 25;         // Raio dos nós
   If Raio < 2 Then Raio := 2;
   If CompArco < 100 Then
      begin
      PenaNormal:=1;
      PenaDiag:=1;
      end;
end;


procedure DesenhaGrafo(ArqNome:String;PaintBox1:TPaintBox);
// Desenha um grafo dado no arquivo ArqNome
var II,x,y:Integer;
    a,Link,Noi,I,J,Nof,I1,J1,d:Integer;
    Simbolo: String[1];
    InicioRota,FimRota: Integer;
    ArcoArtificial: Boolean;
begin
   Form10.Visible:=true;
   AssignFile(Rede,ArqNome);
   Reset(Rede);
   If BotaoAcionado=2 Then        // Se o grafo está contido num arquivo de grafo transformado
      Repeat
      ReadLn(Rede,Simbolo);
      Until Simbolo = '$';

   Read(Rede,M,N,NL,NC,MA,ME);
   i:=0;
   Repeat
      Read(Rede,Simbolo);
      i:=i+1;
   Until (Simbolo='#') or (i>50);
   If Simbolo='#'
      then ReadLn(Rede,InicioRota,FimRota)
      else begin
           ReadLn(Rede);
           InicioRota:=1;
           FimRota:=1;
           end;
   Falha:=false;
```

```
  If ((M=0) Or (M<>MA+ME)) Then
    begin
    Falha:= True;
    ShowMessage('Falha na leitura do Arquivo. Verifique se o arquivo contém um grafo, ou uma solução de
                    MCPP');
    Exit;
    end;
  CalculaCompArco;
  For II:=1 To M Do
    Begin
    ArcoArtificial:=false;
    Readln(Rede,a,Link,Noi,I,J,Nof,I1,J1,d);
    If Link=0
     then CriaAresta(J*CompArco,I*CompArco,J1*CompArco,I1*CompArco,PaintBox1);
    If (II=M) and (Noi=FimRota) and (Nof=InicioRota)
     then ArcoArtificial:=true;
    If (Link=1) and (ArcoArtificial=false)
     then CriaArco(J*CompArco,I*CompArco,J1*CompArco,I1*CompArco,PaintBox1);
    End;
  For I:=1 To NL Do
    For J:=1 To NC Do
      CriaNo(J*CompArco,I*CompArco,PaintBox1);
  CloseFile(Rede);
end;


Procedure OrientaAresta(var f1,f2:Integer;x1,y1,x2,y2:Integer;PaintBox1:TPaintBox);
// Orienta uma aresta no sentido que é percorrido na solução de NegCircuit
// f1 e f2 são a frequencia de travessia da aresta num sentido e no outro
// Se f1>0, a aresta será orientada de inicio para o fim
// Se f1=0, a aresta será orientada no sentido contrário
var xs,ys,xt,yt,temp: Integer;
begin
 If f1=0
 then begin
      temp:=x1;
      x1:=x2;
      x2:=temp;
      temp:=y1;
      y1:=y2;
      y2:=temp;
      f2:=f2-1;
      end
 else f1:=f1-1;
 PaintBox1.Canvas.Pen.Width:=PenaDiag*2;
 PaintBox1.Canvas.Pen.Color:=8400000;
 If (x1 = x2) or (y1 = y2)
    Then PaintBox1.Canvas.Pen.Width:=PenaNormal*2;
 xs:=Round((x2-x1)*0.85)+x1;
 xt:=Round((x2-x1)*0.95)+x1;
 ys:=Round((y2-y1)*0.85)+y1;
 yt:=Round((y2-y1)*0.95)+y1;
 If (x1 = x2) or (y1 = y2) then
   begin
   xs:=Round((x2-x1)*0.80)+x1;
   ys:=Round((y2-y1)*0.80)+y1;
   xt:=Round((x2-x1)*0.90)+x1;
   yt:=Round((y2-y1)*0.90)+y1;
   end;
 PaintBox1.Canvas.MoveTo(xs,ys);
 PaintBox1.Canvas.LineTo(xt,yt);
end;


Procedure  CriaCopia(Var f1,f2: Integer; x1,y1,x2,y2:Integer;PaintBox1:TPaintBox);
// Traça uma cópia de um link com as coordenadas inicio e fim dadas
// f1 e f2 são a frequencia de travessia do links num sentido e no outro
```

```
// Se f1>0, a cópia será orientada de inicio para o fim
// Se f1=0, a cópia será orientada no sentido contrário
var xs,ys,xt,yt,temp,dx,dy,Desloc: Integer;
begin
  If f1=0
  then begin
       temp:=x1;
       x1:=x2;
       x2:=temp;
       temp:=y1;
       y1:=y2;
       y2:=temp;
       f2:=f2-1;
       end
  else f1:=f1-1;
  PaintBox1.Canvas.Pen.Width:=1;
  PaintBox1.Canvas.Pen.Color:=ClRed;

  PaintBox1.Canvas.MoveTo(x1, y1);
  PaintBox1.Canvas.LineTo(x2, y2);

  PaintBox1.Canvas.Pen.Width:=2;
  xs:=Round((x2-x1)*0.85)+x1;
  xt:=Round((x2-x1)*0.95)+x1;
  ys:=Round((y2-y1)*0.85)+y1;
  yt:=Round((y2-y1)*0.95)+y1;
  If (x1 = x2) or (y1 = y2) then
    begin
    xs:=Round((x2-x1)*0.80)+x1;
    ys:=Round((y2-y1)*0.80)+y1;
    xt:=Round((x2-x1)*0.90)+x1;
    yt:=Round((y2-y1)*0.90)+y1;
    end;
  PaintBox1.Canvas.MoveTo(xs,ys);
  PaintBox1.Canvas.LineTo(xt,yt);
end;


Procedure AchaCoordCopiak(k,x1,y1,x2,y2:Integer; var xc1,yc1,xc2,yc2 :Integer);
// Acha as coordenadas de inicio e fim de uma duplicação
var Delta,d:integer;
begin
  Delta:=CompArco div 25;    //Delta: distancia da copia para link original
  If Delta < 5 then Delta:=Delta+1;
  Case k of                  //k: representa a k-ésima que está sendo traçada
    1: Delta := Delta;
    2: Delta := -Delta;
    3: Delta := 2*Delta;
    4: Delta := -2*Delta;
    5: Delta := 3*Delta;
    6: Delta := -3*Delta;
  end;
  d := Round(sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1)+0.0));
  xc1:=Round(x1+0.05*(x2-x1)+Delta*(y2-y1+0.0)/d);
  yc1:=Round(y1+0.05*(y2-y1)-Delta*(x2-x1+0.0)/d);
  xc2:=Round(x1+0.95*(x2-x1)+Delta*(y2-y1+0.0)/d);
  yc2:=Round(y1+0.95*(y2-y1)-Delta*(x2-x1+0.0)/d);

end;


procedure DesenhaGrafoSolucao(ArqNome:String;PaintBox1:TPaintBox);
// Desenha a solução de roteiro, a partir de um arquivo
// de solução gerado na NegCircuit
var II,x,y:Integer;
    a,Link,Noi,I,J,Nof,I1,J1,d,Freq1,Freq2:Integer;
    k,NumCopias,xc1,yc1,xc2,yc2:Integer;
```

```
begin
  AssignFile(Rede,ArqNome);
  Reset(Rede);
  Readln(Rede);
  For II:=1 To M Do
    Begin
    Readln(Rede,a,Link,Noi,I,J,Nof,I1,J1,d,Freq1,Freq2);
    If Link=0
      Then OrientaAresta(Freq1,Freq2,J*CompArco,I*CompArco,J1*CompArco,I1*CompArco,PaintBox1)
      else Freq1:=Freq1-1;
    NumCopias:= Freq1+Freq2;
    If NumCopias >0 Then
      For k := 1 to NumCopias do
        begin
        AchaCoordCopiak(k,J*CompArco,I*CompArco,J1*CompArco,I1*CompArco,xc1,yc1,xc2,yc2);
        CriaCopia(Freq1,Freq2,xc1,yc1,xc2,yc2,PaintBox1);
        end;
    End;
  CloseFile(Rede);
end;


Procedure MovimentaArco(Ordem,x1,y1,x2,y2:Integer;PaintBox1:TPaintBox);
// Cria movimento num link tipo arco
var i,xs,ys,xt,yt,dx,dy,LargSeta: Integer;
   ponto1x,ponto2x,ponto3x,ponto1y,ponto2y,ponto3y:Integer;
begin
  PaintBox1.Canvas.Pen.Width:=PenaDiag + 1;
  PaintBox1.Canvas.Pen.Color:=clBlue;
  If Ordem=3 Then PaintBox1.Canvas.Pen.Color:=ClRed;
  If Ordem=2 Then PaintBox1.Canvas.Pen.Color:=ClYellow;
  If Ordem=1 Then PaintBox1.Canvas.Pen.Color:=clMaroon;
  If ((x1 = x2) or (y1 = y2)) and (BotaoAcionado<>0)
    Then PaintBox1.Canvas.Pen.Width:=PenaNormal+2;
  xs:=x1;
  ys:=y1;
  PaintBox1.Canvas.MoveTo(xs,ys);

  For i:= 1 to 20 do
    begin
    //xt:=Round((x2-x1)*0.05)+xs;
    //yt:=Round((y2-y1)*0.05)+ys;
    xt:=Round((x2-x1)*0.05*i)+x1;
    yt:=Round((y2-y1)*0.05*i)+y1;
    If ((abs(x2-xt)<=abs(Round((x2-x1)*0.05))) and
       (abs(y2-yt)<=abs(Round((y2-y1)*0.05)))) then
      begin
      xt:=x2;
      yt:=y2;
      end;
    PaintBox1.Canvas.LineTo(xt,yt);
    xs:=xt;
    ys:=yt;
    Delay(7*VelInversa);
    end;

  xs:=Round((x2-x1)*0.85)+x1;
  dx:=Round((x2-x1)/Abs(x2-x1+0.1));
  dy:=Round((y2-y1)/Abs(y2-y1+0.1));
  ys:=Round((y2-y1)*0.85)+y1;
  LargSeta:=Round(0.02*Sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1)));
  If dx*dy=0 Then
    begin
    xs:=Round((x2-x1)*0.80)+x1;
    ys:=Round((y2-y1)*0.80)+y1;
    LargSeta:=2*LargSeta;
    end;
```

```
  Ponto1x:=xs-dy*LargSeta;
  Ponto1y:=ys+dx*LargSeta;
  Ponto2x:=xs+dy*LargSeta;
  Ponto2y:=ys-dx*LargSeta;
  Ponto3x:=Round((x2-x1)*0.93)+x1;
  Ponto3y:=Round((y2-y1)*0.93)+y1;
  If dx*dy=0 Then
    begin
    Ponto3x:=Round((x2-x1)*0.90)+x1;
    Ponto3y:=Round((y2-y1)*0.90)+y1;
    end;

  PaintBox1.Canvas.Pen.Width:=PenaDiag;
  If (x1 = x2) or (y1 = y2)
     Then PaintBox1.Canvas.Pen.Width:=PenaNormal;
  PaintBox1.Canvas.Pen.Color:=clMaroon;
  PaintBox1.Canvas.Polygon([point(ponto1x,ponto1y),point(ponto2x,ponto2y),
     point(ponto3x,ponto3y)]);
end;


Procedure MovimentaAresta(Ordem,x1,y1,x2,y2:Integer;PaintBox1:TPaintBox);
// Cria movimento num link tipo aresta
var i,xs,ys,xt,yt,temp: Integer;
begin
  PaintBox1.Canvas.Pen.Width:=PenaDiag+1;
  PaintBox1.Canvas.Pen.Color:=clBlue;
  If Ordem=3 Then PaintBox1.Canvas.Pen.Color:=ClRed;
  If Ordem=2 Then PaintBox1.Canvas.Pen.Color:=ClYellow;
  If Ordem=1 Then PaintBox1.Canvas.Pen.Color:=clMaroon;
  If ((x1 = x2) or (y1 = y2)) and (BotaoAcionado<>0)
     Then PaintBox1.Canvas.Pen.Width:=PenaNormal+2;
  xs:=x1;
  ys:=y1;
  PaintBox1.Canvas.MoveTo(xs,ys);

  For i:= 1 to 20 do
     begin
     //xt:=Round((x2-x1)*0.05)+xs;
     //yt:=Round((y2-y1)*0.05)+ys;
     xt:=Round((x2-x1)*0.05*i)+x1;
     yt:=Round((y2-y1)*0.05*i)+y1;
     If ((abs(x2-xt)<=abs(Round((x2-x1)*0.05))) and
        (abs(y2-yt)<=abs(Round((y2-y1)*0.05)))) then
        begin
        xt:=x2;
        yt:=y2;
        end;
     PaintBox1.Canvas.LineTo(xt,yt);
     xs:=xt;
     ys:=yt;
     Delay(7*VelInversa);
     end;
end;


Procedure MovimentaCopia(Ordem,FreqTotal,x1,y1,x2,y2:Integer;PaintBox1:TPaintBox);
// Cria movimento num link tipo cópia
var i,xs,ys,xt,yt,temp: Integer;
begin
  PaintBox1.Canvas.Pen.Width:=PenaDiag+1;
  PaintBox1.Canvas.Pen.Color:=clBlue;
  If Ordem=3 Then PaintBox1.Canvas.Pen.Color:=ClRed;
  If Ordem=2 Then PaintBox1.Canvas.Pen.Color:=ClYellow;
  If Ordem=1 Then PaintBox1.Canvas.Pen.Color:=clMaroon;
  If ((x1 = x2) or (y1 = y2)) and (BotaoAcionado<>0)
     Then PaintBox1.Canvas.Pen.Width:=PenaNormal+2;
```

```
  xs:=x1;
  ys:=y1;
  PaintBox1.Canvas.MoveTo(xs,ys);
  For i:= 1 to 20 do
     begin
     //xt:=Round((x2-x1)*0.05)+xs;
     //yt:=Round((y2-y1)*0.05)+ys;
     xt:=Round((x2-x1)*0.05*i)+x1;
     yt:=Round((y2-y1)*0.05*i)+y1;
     If ((abs(x2-xt)<=abs(Round((x2-x1)*0.05))) and
        (abs(y2-yt)<=abs(Round((y2-y1)*0.05)))) then
        begin
        xt:=x2;
        yt:=y2;
        end;
     PaintBox1.Canvas.LineTo(xt,yt);
     xs:=xt;
     ys:=yt;
     Delay(7*VelInversa);
     end;
end;


procedure DesenhaGrafoSolucaoGLS(ArqNome,ArqSol:String;PaintBox1:TPaintBox);
// Desenha a solução de roteiro, a partir de um arquivo
// de solução de GLS, gerado na UPrincipal
var II,x,y,x1,y1,x2,y2,Freq1,Freq2,Ordem:Integer;
    a,Link,Noi,I,J,Nof,I1,J1,I2,J2,d,Sentido:Integer;
    k,NumCopias,xc1,yc1,xc2,yc2,NumLink,NumLinkCad:Integer;
    InicioRota, FimRota: Integer;
    Tipo,Simbolo: String[1];
    Arq: Array[1..MaxLink] of TLink;
    Rota: Array[1..MaxRota] of TRota;
    RegTemp: TRota;
    No: Array[1..MaxNo] of TNo;
    Demora:Real;
begin
   AssignFile(Rede,ArqNome);    //Arquivo Grafo Transformado
   Reset(Rede);
   Repeat
     ReadLn(Rede,Simbolo);
   Until Simbolo = '$';
   Read(Rede,M,N,NL,NC,MA,ME);
   i:=0;
   Repeat
      Read(Rede,Simbolo);       // lendo os nós Inicio e Fim da Rota
      i:=i+1;
   Until (Simbolo='#') or (i>50);
   If Simbolo='#'
      then ReadLn(Rede,InicioRota,FimRota)
      else begin
           ReadLn(Rede);
           InicioRota:=1;
           FimRota:=1;
           end;
   For i:= 1 to N do
     begin
     No[i].ci:= 0;
     No[i].cj:= 0;
     end;
   For i:= 1 to M do            //Carrega vetor Arq de links junto
     begin                    //com as coordenadas dos nos
     Readln(Rede,a,Arq[i].Tipo,Arq[i].Noi,I1,J1,
          Arq[i].Nof,I2,J2,d);
     Noi:= Arq[i].Noi;
     Nof:= Arq[i].Nof;
     If No[Noi].ci = 0 Then
```

```
    begin
    No[Noi].ci := I1;
    No[Noi].cj := J1;
    end;
  If No[Nof].ci = 0 Then
    begin
    No[Nof].ci := I2;
    No[Nof].cj := J2;
    end;
  end;
CloseFile(Rede);

AssignFile(Roteiro,ArqSol);  //Arquivo solução
Reset(Roteiro);
For II := 1 To 12 do
   Readln(Roteiro,Tipo);     //
II:=0;
While Not EoF (Roteiro) do
   Begin
   II:=II+1;;
   Read(Roteiro,Noi);
   Read(Roteiro,Nof);
   Repeat Read(Roteiro,Tipo) Until Tipo > ' ';
   Readln(Roteiro,a);
   Rota[II].Arc := a;        //Rota é o vetor que contem a
   Rota[II].Tipo:=Tipo;      //sequencia dos links na solucao
   Rota[II].Noi:=Noi;        //Tipo= + :aresta percorrido no sentido normal
                             //      - :aresta percorrido no sentido contrário
                             //      > :arco
                             //      D :trecho duplicado que pode ser uma cadeia
   Rota[II].Nof:=Nof;
   Rota[II].Ordem:=0;        // numero das vezes que até o momento o link apareceu no roteiro
   Rota[II].FreqTotal:=0;    // numero total das vezes que o link é percorrido no roteiro
   end;
Rota[II+1] := Rota[1];
CloseFile(Roteiro);
NumLink:=II;                 // NumLink: numero total de link / cadeias duplicadas

II:=0;
If InicioRota<>FimRota
 Then
    begin
    Repeat                    // procurando o link que começa depois
      II:=II+1;               // do Arco artificial (FimRota, InicioRota)
      If II>NumLink then Exit;
      Rota[NumLink+II]:=Rota[II];
    Until (Rota[II].Noi = FimRota) and
          (Rota[II].Nof = InicioRota) and
          (Rota[II].Tipo = '>');
    NumLink:=NumLink-1;          // Excluindo o arco artificial
    For i:= 1 to NumLink do      // Reordenando a rota, para começar do nó InicioRota
      Rota[i]:=Rota[i+II];
    end
 else
    begin
    Repeat                    // procurando o link que começa com o nó InicioRota
      II:=II+1;
      If II>NumLink then Exit;
      Rota[NumLink+II]:=Rota[II];
    Until Rota[II].Noi = InicioRota;
    For i:= 1 to NumLink do      // Reordenando a rota, para começar do nó InicioRota
      Rota[i]:=Rota[i+II-1];
    end;


For II := 1 to NumLink do    // tracando um trecho duplicado pelo link original
   begin                     // de modo que primeiro o trecho original seja percorrido
```

```
  If ((Rota[II].Tipo = 'D') and (II < NumLink)) then
    begin
    Noi:=Rota[II].Noi;
    Nof:=Rota[II].Nof;
    For i:=II+1 to NumLink do
      If ((Noi=Rota[i].Noi) and (Nof=Rota[i].Nof))
        then If Rota[i].Tipo <> 'D'
          then begin
              RegTemp:= Rota[II];
              Rota[II]:=Rota[i];
              Rota[i]:=RegTemp;
              end;
    end;
  end;

For II := 1 To NumLink do
  begin                  // procurando ver quantas vezes cada link aparece no roteiro
  Noi:=Rota[II].Noi;
  Nof:=Rota[II].Nof;
  For i:=1 to II do
    begin
    If (((Noi=Rota[i].Noi) and (Nof=Rota[i].Nof)) or
       ((Noi=Rota[i].Nof) and (Nof=Rota[i].Noi)))
     then begin
          Rota[II].FreqTotal := Rota[II].FreqTotal +1;
          If i<>II Then Rota[i].FreqTotal := Rota[i].FreqTotal +1;
          end;
    end;
  Rota[II].Ordem := Rota[II].FreqTotal;
  end;

Velocidade := Form1.SpinEdit1.Value;
VelInversa:= 4000 div Velocidade;       // 4000: constante para calibragem da velocidade
Demora:=Delay(10000);                   // demora inicial para começar o movimento
For II := 1 To NumLink do               // Movimentar cada link na sequencia do roteiro
  begin
  x1:= No[Rota[II].Noi].cj*CompArco;    // coordenada x do Noi
  y1:= No[Rota[II].Noi].ci*CompArco;    // coordenada y do Noi
  x2:= No[Rota[II].Nof].cj*CompArco;    // coordenada x do Nof
  y2:= No[Rota[II].Nof].ci*CompArco;    // coordenada y do Nof
  If ((Rota[II].Tipo = '+') Or (Rota[II].Tipo = '-'))
    Then MovimentaAresta(Rota[II].Ordem,x1,y1,x2,y2,PaintBox1);
  If Rota[II].Tipo = '>'
    Then MovimentaArco(Rota[II].Ordem,x1,y1,x2,y2,PaintBox1);
  If Rota[II].Tipo = 'D'
    Then If Rota[II].FreqTotal > 1
      Then MovimentaCopia(Rota[II].Ordem,Rota[II].FreqTotal,x1,y1,x2,y2,PaintBox1)
      Else begin
           Freq1:=1;
           Freq2:=0;
           AchaCoordCopiak(Freq1,x1,y1,x2,y2,xc1,yc1,xc2,yc2);
           Delay(50*VelInversa);
           CriaCopia(Freq1,Freq2,xc1,yc1,xc2,yc2,PaintBox1);
           Delay(50*VelInversa);
           end;
  Freq1:=1;
  Freq2:=0;
  Ordem:= Rota[II].Ordem-1;
  If Ordem > 0 Then
    begin
    AchaCoordCopiak(Ordem,x1,y1,x2,y2,xc1,yc1,xc2,yc2);
    CriaCopia(Freq1,Freq2,xc1,yc1,xc2,yc2,PaintBox1);
    end;
  end;

For I:=1 To NL Do           // redesenha os nós
  For J:=1 To NC Do
```

```
        CriaNo(J*CompArco,I*CompArco,PaintBox1);
end;


Procedure DesenhaGrafoSolucaoMapa(ArqMapa,ArqNome,ArqSol: String; PaintBox1: TPaintBox);
// Desenha a solução de roteiro, a partir de um arquivo
// de solução de GLS, gerado na UPrincipal
var II,x,y,x1,y1,x2,y2,Freq1,Freq2,Ordem:Integer;
    a,Link,Noi,I,J,Nof,I1,J1,I2,J2,d,Sentido:Integer;
    k,NumCopias,xc1,yc1,xc2,yc2,NumLink,NumLinkCad:Integer;
    InicioRota, FimRota: Integer;
    Tipo,Simbolo: String[1];
    Arq: Array[1..MaxLink] of TLink;
    Rota: Array[1..MaxRota] of TRota;
    RegTemp: TRota;
    No: Array[1..MaxNo] of TNo;
    Demora:Real;
begin
   CompArco:=1;
   Form10.Visible:=true;
   Form10.Image1.Visible:=true;
   Form10.OpenPictureDialog1.FileName:=ArqMapa;
   Form10.Image1.Picture.LoadFromFile(Form10.OpenPictureDialog1.FileName);
   Form10.Refresh;
   AssignFile(Rede,ArqNome);    //Arquivo Grafo Transformado
   Reset(Rede);
   Repeat
     ReadLn(Rede,Simbolo);
   Until Simbolo = '$';

   Read(Rede,M,N,NL,NC,MA,ME);
   i:=0;
   Repeat
     Read(Rede,Simbolo);       // lendo os nós Inicio e Fim da Rota
     i:=i+1;
   Until (Simbolo='#') or (i>50);
   If Simbolo='#'
     then ReadLn(Rede,InicioRota,FimRota)
     else begin
          ReadLn(Rede);
          InicioRota:=1;
          FimRota:=1;
          end;
   For i:= 1 to N do
    begin
    No[i].ci:= 0;
    No[i].cj:= 0;
    end;
   For i:= 1 to M do           //Carrega vetor Arq de links junto
    begin                    //com as coordenadas dos nos
    Readln(Rede,a,Arq[i].Tipo,Arq[i].Noi,I1,J1,
          Arq[i].Nof,I2,J2,d);
    Noi:= Arq[i].Noi;
    Nof:= Arq[i].Nof;
    If No[Noi].ci = 0 Then
      begin
      No[Noi].ci := I1;
      No[Noi].cj := J1;
      end;
    If No[Nof].ci = 0 Then
      begin
      No[Nof].ci := I2;
      No[Nof].cj := J2;
      end;
    end;
   CloseFile(Rede);
```

```
AssignFile(Roteiro,ArqSol);  //Arquivo solução
Reset(Roteiro);
For II := 1 To 12 do
  Readln(Roteiro,Tipo);     //
II:=0;
While Not EoF (Roteiro) do
  Begin
  II:=II+1;;
  Read(Roteiro,Noi);
  Read(Roteiro,Nof);
  Repeat Read(Roteiro,Tipo) Until Tipo > ' ';
  Readln(Roteiro,a);
  Rota[II].Arc := a;        //Rota é o vetor que contem a
  Rota[II].Tipo:=Tipo;      //sequencia dos links na solucao
  Rota[II].Noi:=Noi;        //Tipo= + :aresta percorrido no sentido normal
                            //      - :aresta percorrido no sentido contrário
                            //      > :arco
                            //      D :trecho duplicado que pode ser uma cadeia
  Rota[II].Nof:=Nof;
  Rota[II].Ordem:=0;        // numero das vezes que até o momento o link apareceu no roteiro
  Rota[II].FreqTotal:=0;    // numero total das vezes que o link é percorrido no roteiro
  end;
Rota[II+1] := Rota[1];
CloseFile(Roteiro);
NumLink:=II;                // NumLink: numero total de link / cadeias duplicadas

II:=0;
If InicioRota<>FimRota
 Then
   begin
   Repeat                    // procurando o link que começa depois
     II:=II+1;               // do Arco artificial (FimRota, InicioRota)
     If II>NumLink then Exit;
     Rota[NumLink+II]:=Rota[II];
   Until (Rota[II].Noi = FimRota) and
         (Rota[II].Nof = InicioRota) and
         (Rota[II].Tipo = '>');
   NumLink:=NumLink-1;          // Excluindo o arco artificial
   For i:= 1 to NumLink do      // Reordenando a rota, para começar do nó InicioRota
     Rota[i]:=Rota[i+II];
   end
 else
   begin
   Repeat                    // procurando o link que começa com o nó InicioRota
     II:=II+1;
     If II>NumLink then Exit;
     Rota[NumLink+II]:=Rota[II];
   Until Rota[II].Noi = InicioRota;
   For i:= 1 to NumLink do      // Reordenando a rota, para começar do nó InicioRota
     Rota[i]:=Rota[i+II-1];
   end;

For II := 1 to NumLink do    // tracando um trecho duplicado pelo link original
  begin                      // de modo que primeiro o trecho original seja percorrido
  If ((Rota[II].Tipo = 'D') and (II < NumLink)) then
    begin
    Noi:=Rota[II].Noi;
    Nof:=Rota[II].Nof;
    For i:=II+1 to NumLink do
      If ((Noi=Rota[i].Noi) and (Nof=Rota[i].Nof))
        then If Rota[i].Tipo <> 'D'
          then begin
               RegTemp:= Rota[II];
               Rota[II]:=Rota[i];
               Rota[i]:=RegTemp;
               end;
    end;
```

```
    end;

  For II := 1 To NumLink do
    begin                  // procurando ver quantas vezes cada link aparece no roteiro
    Noi:=Rota[II].Noi;
    Nof:=Rota[II].Nof;
    For i:=1 to II do
      begin
      If (((Noi=Rota[i].Noi) and (Nof=Rota[i].Nof)) or
          ((Noi=Rota[i].Nof) and (Nof=Rota[i].Noi)))
        then begin
            Rota[II].FreqTotal := Rota[II].FreqTotal +1;
            If i<>II Then Rota[i].FreqTotal := Rota[i].FreqTotal +1;
            end;
      end;
    Rota[II].Ordem := Rota[II].FreqTotal;
    end;

  Velocidade := Form1.SpinEdit1.Value;
  VelInversa:= 2000 div Velocidade;       // 1000: constante para calibragem da velocidade
  Demora:=Delay(10000);                  // demora inicial para começar o movimento
  For II := 1 To NumLink do              // Movimentar cada link na sequencia do roteiro
    begin
    x1:= No[Rota[II].Noi].cj*CompArco;    // coordenada x do Noi
    y1:= No[Rota[II].Noi].ci*CompArco;    // coordenada y do Noi
    x2:= No[Rota[II].Nof].cj*CompArco;    // coordenada x do Nof
    y2:= No[Rota[II].Nof].ci*CompArco;    // coordenada y do Nof
    If ((Rota[II].Tipo = '+') Or (Rota[II].Tipo = '-'))
      Then MovimentaAresta(Rota[II].Ordem,x1,y1,x2,y2,PaintBox1);
    If Rota[II].Tipo = '>'
      Then MovimentaArco(Rota[II].Ordem,x1,y1,x2,y2,PaintBox1);
    If Rota[II].Tipo = 'D'
      Then If Rota[II].FreqTotal > 1
        Then MovimentaCopia(Rota[II].Ordem,Rota[II].FreqTotal,x1,y1,x2,y2,PaintBox1)
        Else begin
            Freq1:=1;
            Freq2:=0;
            AchaCoordCopiak(Freq1,x1,y1,x2,y2,xc1,yc1,xc2,yc2);
            Delay(50*VelInversa);
            CriaCopia(Freq1,Freq2,xc1,yc1,xc2,yc2,PaintBox1);
            Delay(50*VelInversa);
            end;
    Freq1:=1;
    Freq2:=0;
    Ordem:= Rota[II].Ordem-1;
    If Ordem > 0 Then
      begin
      AchaCoordCopiak(Ordem,x1,y1,x2,y2,xc1,yc1,xc2,yc2);
      CriaCopia(Freq1,Freq2,xc1,yc1,xc2,yc2,PaintBox1);
      end;
    end;
end;


procedure TForm10.Button1Click(Sender: TObject);
// Desenha Grafo Gerado por URandGraf
begin
  Form10.Image1.Visible:=false;
  BotaoAcionado:=1;
  ArqNome:=Form1.SaveDialog1.FileName+'.Txt';
  //ArqNome:=Form1.SaveDialog1.FileName;
  DesenhaGrafo(ArqNome,PaintBox1);
end;


procedure TForm10.Button2Click(Sender: TObject);
// Desenha Solução GLS (UPrincipal)
```

```
begin
  Form10.Image1.Visible:=false;
  If Form1.CheckBox5.Checked = false Then Exit;
  BotaoAcionado:=2;
  ArqNome:= FrmPrincipal.OpenDialog1.FileName;
  DesenhaGrafo(ArqNome,PaintBox1);
  ArqSol:= FrmPrincipal.SaveDialog2.FileName;
  DesenhaGrafoSolucaoGLS(ArqNome,ArqSol,PaintBox1);
end;


procedure TForm10.Button3Click(Sender: TObject);
// Desenha Solução da NegCircuit
begin
  Form10.Image1.Visible:=false;
  If Form1.CheckBox5.Checked = false Then Exit;
  BotaoAcionado:=3;
  ArqNome:= Form4.SaveDialog1.FileName;
  DesenhaGrafo(ArqNome,PaintBox1);
  DesenhaGrafoSolucao(ArqNome,PaintBox1);

end;


procedure TForm10.Button4Click(Sender: TObject);
// Esta rotina identifica o tipo de arquivo, se ele é um grafo,
// se ele é uma solução de MCPP gerada pela rotina NegCircuit,
// ou se é uma solução de MCPP gerada pela Busca Local.
// Depois da identificação, desenha o grafo e mostra a solução.
var St:String;
begin
  BotaoAcionado:=4;
  Form10.Image1.Visible:=false;
  //OpenDialog1.InitialDir := 'C:\Arquivos de programas\Borland\Delphi4\Projects\DadosMcpp\';
  OpenDialog1.InitialDir := 'C:\Meus documentos\Doutorado\MCPPgls Demo\';
  If OpenDialog1.Execute
    Then ArqNome := OpenDialog1.FileName
    Else Exit;
  St:=Copy(ArqNome,Length(ArqNome)-6,3);
  If (St='KXS') or (St='KUS') or (St='KCS') or (St='KNS')
        then begin
            // Anima solução sobre um mapa
            BotaoAcionado:=0;
            CalculaCompArco;
            ArqSol:=ArqNome;
            ArqMapa:=ArqNome;
            Delete (ArqNome,Length(ArqNome)-4,1);
            Delete (ArqMapa,Length(ArqMapa)-6,7);
            ArqMapa:=ArqMapa+'.bmp';
            DesenhaGrafoSolucaoMapa(ArqMapa,ArqNome,ArqSol,PaintBox1);
            Exit;
            end;
  If Copy(ArqNome,Length(ArqNome)-4,1)<>'S'
    then begin
        If Copy(ArqNome,Length(ArqNome)-4,1)='L'
          then begin
            // Desenha solução de Lower Bound
            BotaoAcionado:=3;
            DesenhaGrafo(ArqNome,PaintBox1);
            DesenhaGrafoSolucao(ArqNome,PaintBox1);
            end
          else begin
            // Desenha um grafo
            BotaoAcionado:=1;
            DesenhaGrafo(ArqNome,PaintBox1)
            end;
        end
```

```
    else begin
        St:= Copy(ArqNome,Length(ArqNome)-5,1);
        If ((St = 'X') Or (St = 'U') Or (St = 'C') Or (St = 'N'))
          then begin
              // Anima solução de Busca Local
              ArqSol:=ArqNome;
              Delete (ArqNome,Length(ArqNome)-4,1);
              BotaoAcionado:=2;
              DesenhaGrafo(ArqNome,PaintBox1);
              DesenhaGrafoSolucaoGLS(ArqNome,ArqSol,PaintBox1);
              end
          else begin
              // Desenha solução de NegCircuit
              BotaoAcionado:=3;
              DesenhaGrafo(ArqNome,PaintBox1);
              DesenhaGrafoSolucao(ArqNome,PaintBox1);
              end;
        end;
end;


procedure TForm10.PaintBox1Paint(Sender: TObject);
begin
  If falha then Exit;
  If BotaoAcionado=1 then DesenhaGrafo(ArqNome,PaintBox1);
  //If BotaoAcionado=2 then Form10.Button2.Click;
  If BotaoAcionado=3
    then begin
        DesenhaGrafo(ArqNome,PaintBox1);
        DesenhaGrafoSolucao(ArqNome,PaintBox1);
        end;
end;

procedure TForm10.Button5Click(Sender: TObject);
// Anima solução sobre um mapa
// Esta rotina é chamada da UPrincipal
begin
   If Form1.CheckBox5.Checked = false Then Exit;
   ArqNome:= FrmPrincipal.OpenDialog1.FileName;
   ArqSol:= FrmPrincipal.SaveDialog2.FileName;
   BotaoAcionado:=0;
   CalculaCompArco;
   ArqMapa:=ArqSol;
   Delete (ArqMapa,Length(ArqMapa)-6,7);
   ArqMapa:=ArqMapa+'.bmp';
   DesenhaGrafoSolucaoMapa(ArqMapa,ArqNome,ArqSol,PaintBox1);
end;

end.
```