UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

UM MODELO DE RESOLUÇÃO PARA O PROBLEMA DE ROTEIRIZAÇÃO EM ARCOS COM RESTRIÇÃO DE CAPACIDADE

Rafael Roco de Araújo

Porto Alegre, 2003

Rafael Roco de Araújo

UM MODELO DE RESOLUÇÃO PARA O PROBLEMA DE ROTEIRIZAÇÃO EM ARCOS COM RESTRIÇÃO DE CAPACIDADE

Dissertação submetida ao Programa de Pós-Graduação em Engenharia de Produção da Universidade Federal do Rio Grande do Sul como requisito parcial à obtenção do título de MESTRE EM ENGENHARIA EM ENGENHARIA DE PRODUÇÃO na área de concentração de Sistemas de Transportes e Logística.

Orientador: Prof.Luiz Afonso dos Santos Senna, Ph. D

Porto Alegre, 2003

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Engenharia de Produção e aprovada em sua forma final pelo Orientador e pela Banca Examinadora designada pelo Programa de Pós-Graduação em Engenharia de Produção.

Prof. Luiz Afonso dos Santos Senna, Ph. D
Universidade Federal do Rio Grande do Sul

Prof. José Luiz Duarte Ribeiro, Ph.D
Coordenador PPGEP / UFRGS

Banca Examinadora:

Ana Maria Volkmer de Azambuja, Dr.
Prof. Depto. Matemática / FURG

Emílio Merino Dominguez, Dr.
Prof. PPGEP/UFRGS

Fernando Dutra Michel, M. Eng.
Prof. PPGEP/UFRGS

João Luiz Becker, Ph. D
Prof. PPGA/UFRGS

A meus pais, Arno e Vitória, e a memória de meu tio Rafael Rocco.

AGRADECIMENTOS

Vários são aqueles que colaboraram de algum modo, seja de forma direta ou indireta, para o desenvolvimento deste trabalho.

Gostaria de registrar o meu sincero agradecimento ao Prof. Fernando Michel pelo permanente acompanhamento, pelo apoio e pela amizade, assim como ao meu orientador o Prof. Luiz Afonso dos Santos Senna.

Agradeço também aos colegas Guilherme Schneider e Daniela Facchini pela valiosa participação que tiveram no desenvolvimento deste trabalho como bolsistas de iniciação científica.

A todas as pessoas que atuam no Lastran, sejam elas professores, doutorandos, mestrandos, bolsistas e funcionários, agradeço pela amizade e constante incentivo.

Agradeço ao suporte financeiro da FEENG durante este período e também ao PPGEP por toda a estrutura de apoio que nos foi disponibilizada.

Aos meus familiares, em especial a meu pai, o agradecimento por todo apoio e encorajamento, decisivo principalmente para a superação dos momentos mais difíceis.

Por fim, rendo graças a Deus, por todos os caminhos que pelas suas sábias mãos foram abertos neste importante período de minha vida, assim como pela sua constante benção e proteção.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE SÍMBOLOS

2 – Problemas de Roteirização em Arcos

Alfabeto grego

δ_i – conjunto das arestas incidentes no vértice v_i

λ - penalidade associada as arestas em um grafo condensado na preparação para resolução do PCR

Alfabeto latim caixa baixa

a_{ij} – arco compreendido entre os vértices v_i e v_j.

c – quantidade de classes de conjunto de arestas com uma relação de precedência

c_{ij} – coeficiente de custo do segmento compreendido entre os vértices v_i e v_j

d_j – número de arcos partindo de um vértice v_j que excede o número de arcos chegando a v_j

e_{ij} – aresta compreendida entre os vértices v_i e v_j.

g_i – grau de um vértice de índice i

k - número de unidades de trabalho

l_{ij}^{p} - variável binária que assume o valor 1 se a unidade p atende o segmento compreendido entre v_i e v_j e 0 caso contrário.

m – número de arestas existentes em E

n – e-néssimo elemento

q_{ij} - demanda existente no segmento compreendido entre os vértices v_i e v_j

p – unidade de transporte

r – conjunto dos caminhos mínimos entre pares de vértices de grau ímpar

s – primeiro vértice de grau ímpar existente em um grafo

s_i – número de arcos chegando a um vértice v_i que excede o número de arcos partindo de v_i

t – segundo vértice de grau ímpar existente em um grafo

v_D – base operacional de grupo de uma frota ou equipe

v_i – vértice de índice i

v_n – e-néssimo vértice

v_r – vértice raiz de uma arborescência

v_x – vértice de índice x

v_y – vértice de índice y

x_{ij} – número de vezes que o segmento compreendido entre os vértices v_i e v_j é percorrido

x_{ij}^{p} - uma variável binária que assume o valor 1 se o segmento compreendido entre v_i e v_j é atravessado pela unidade p e 0 caso contrário

Alfabeto latim caixa alta

A – conjunto de arcos contidos em um grafo

A'- conjunto aumentado de arcos

A_R – conjunto de arcos que demanda serviço

C_R – componentes desconexas orientadas

C_T – custo total de um percurso

C_x – componente desconexa de índice x

C_y – componente desconexa de índice y
G – grafo que representa uma rede viária
G' - grafo aumentado
G_c – grafo condensado
G_R – grafo aumentado formado por $G \cup E_T$
E- conjunto de arestas contidas em um grafo
E'- conjunto aumentado de arestas
E_c – conjunto de arestas de classe c
E_M – conjunto das arestas geradas a partir da resolução de um problema de emparelhamento perfeito de peso mínimo sobre $G \cup E_T$
E_R – conjunto das arestas que demandam serviço
E_T – conjunto de arestas geradas a partir da resolução de um problema de árvore de peso mínimo
E_{To} - conjunto inicial de arestas geradas a partir da resolução de um problema de emparelhamento perfeito de peso mínimo no processo de preparação para resolução do PCR
F'- conjunto de arestas que representam os caminhos mínimos entre os vértices de grau ímpar
I – conjunto de vértices v_i
J – conjunto de vértices v_j
M' - conjunto de arestas que formam um emparelhamento perfeito em R_T
O – ordem de complexidade de um algoritmo
R_B – grafo bipartido
R_T – grafo completo
V_e – conjunto dos vértices de grau par
V_o – conjunto dos vértices de grau ímpar
W – capacidade máxima de uma unidade de trabalho

3 – Problemas de Particionamento de Redes de Transporte

Alfabeto latim caixa baixa

n – valor limite da restrição de uma unidade de trabalho
k – número de partições
k_i – limite inferior para o número de partições
k_S - limite superior para o número de partições
q_{ij} - demanda existente no segmento compreendido entre os vértices v_i e v_j

Alfabeto latim caixa alta

C_T - demanda total existente em uma rede de atendimento considerando os trechos de percurso efetivo
C_{TS} - tempo total de atendimento da rede considerando o somatório dos trechos de percurso efetivo e dos trechos de percurso ocioso.
G – grafo que representa uma rede viária
E- conjunto de arestas contidas em um grafo
SG_i – partição de índice i
SG_j – partição de índice j
SG_k – k-éssima partição
V – conjunto dos vértices existentes em um grafo

4 - Proposição de um Modelo para Resolução do Problema de Roteirização em Arcos Capacitado

Alfabeto grego

λ - penalidade associada as arestas em um grafo condensado na preparação para resolução do PCR

Alfabeto latim caixa baixa

c_{ij} – coeficiente de custo que representa o tempo de percurso efetivo (com trabalho) do segmento compreendido entre os vértices v_i e v_j.
c'_{ij} – coeficiente de custo que representa o tempo de percurso ocioso (sem trabalho) do segmento compreendido entre os vértices v_i e v_j.
e_{ij} – aresta compreendida entre os vértices v_i e v_j.
e_p – aresta promissora.
e_{xy} – aresta compreendida entre os vértices v_x e v_y.
g_i - grau de incidência de um vértice.
k – número de partições.
n – valor limite da restrição de tempo de jornada de trabalho.
tv_p – lista que armazena os candidatos a vértice-pai.
v_f - vértice-filho.
v_i – vértice de índice i
v_D – vértice-depósito.
v_j – vértice de índice j
v_p – vértice-pai
v_x – vértice de índice x
v_y – vértice de índice y

Alfabeto latim caixa alta

C – conjunto que representa como vértices as componentes desconexas existentes em um partição
C_c – grafo condensado completo
C_T – custo total para o atendimento das demandas existentes em uma rede
C_{TSG} – custo total para o atendimento das demandas existentes em uma partição
C_{TSGi} – custo total para o atendimento das demandas existentes em uma partição de índice i
C_{TSGj} – custo total para o atendimento das demandas existentes em uma partição de índice j
C_x – componente desconexa de índice x
C_y – componente desconexa de índice y
E- conjunto de arestas contidas em um grafo
E_M – conjunto das arestas geradas a partir da resolução de um problema de emparelhamento perfeito de peso mínimo sobre $G \cup E_T$
E_R – conjunto das arestas que demandam serviço
E_T – conjunto de arestas geradas a partir da resolução de um problema de árvore de peso mínimo
E_{To} - conjunto inicial de arestas geradas a partir da resolução de um problema de emparelhamento perfeito de peso mínimo no processo de preparação para resolução do PCR
G – grafo que representa uma rede viária
P_{ij} – ganho no custo resultante da troca de uma aresta promissora de uma partição para outra
R_{ij} – ganho total da troca de uma aresta promissora de uma partição para outra
R_T – grafo completo

SG – partição
SG'- partição aumentada
SG_c – partição condensada
SG_i – partição de índice i
SG_j – partição de índice j
T_{ij} – ganho no fator de forma
V – conjunto dos vértices existentes em um grafo
V_i – número de vértice existentes em uma partição de índice i antes da troca de uma aresta promissora
V'_i – número de vértice existentes em uma partição de índice i depois da troca de uma aresta promissora
V_j – número de vértice existentes em uma partição de índice j antes da troca de uma aresta promissora
V'_j – número de vértice existentes em uma partição de índice j depois da troca de uma aresta promissora

SUMÁRIO

RESUMO

Os Problemas de Roteirização em Arcos constituem uma classe dos Problemas de Roteirização de Veículos, nos quais as demandas ocorrem de maneira contínua ao longo dos segmentos de uma rede de transporte. O campo de aplicação dos Problemas de Roteirização em Arcos abrange o planejamento da operação de serviços como a coleta de lixo; a entrega de correspondência e de jornais; a leitura de medidores de energia elétrica, água e gás; e o transporte escolar por ônibus. Nas situações de ordem prática é comum a ocorrência de restrições operacionais que impossibilitam o atendimento de todas as demandas existentes em uma rede de transporte por um único veículo ou pessoa. Desse modo, além da seqüência de percurso, é necessário determinar o conjunto de segmentos de via que cada unidade de trabalho pertencente a uma frota ou equipe de serviço deve atender. Este trabalho apresenta os aspectos conceituais e resolutivos dos Problemas de Roteirização em Arcos e do Problema do Particionamento de Redes de Transporte. É feita também a proposição de um modelo de resolução, desenvolvido a partir da estratégia de agrupar primeiro – roteirizar depois, para o Problema de Roteirização em Arcos Capacitado descrito em grafos não orientados. A demonstração do modelo é feita através de sua aplicação em um estudo de caso. São feitas, a seguir, a análise do desempenho do modelo nesta aplicação e a comparação entre os resultados obtidos e a prática atualmente utilizada, a qual é baseada na experiência de um planejador.

Palavras-chave: roteirização em arcos; particionamento de redes; modelos de resolução.

ABSTRACT

Arc Routing Problems constitute a Vehicle Routing Problems class where demands occur in a continuous fashion along of the transportation network segments. The application field of the Arc Routing Problems covers operation planning of services as garbage collection; mail and newspaper delivery; electric, water and gas meter reading; and school bus transportation. In practical situations are common operational constraints occurrence that make impossible the satisfaction of all demands in a transportation network by a single vehicle or person. In this way, besides of tour sequence, it is necessary to determine the road segments that each work unity of a fleet or crew service must serve. This work presents the conceptual and resolution aspects of the Arc Routing Problems and the Transportation Network Partitioning Problem. It is also proposed a resolution model, developed from the cluster first – route second strategy, for the Capacitated Arc Routing Problem described in not oriented graphs. The model demonstration is done by its application in a case study. In sequence, it is conducted the analysis of the model performance in this application and the comparison between the obtained results and the practice presently used, which is based on the experience of a planner.

Key-words: arc routing; network partitioning; resolution models.

1 INTRODUÇÃO

1.1 O CONTEXTO DOS PROBLEMAS DE ROTEIRIZAÇÃO NOS SISTEMAS LOGÍSTICOS

O atual cenário do setor produtivo tem exigido das organizações elevados padrões de produção, com ênfase no nível de serviço prestado aos clientes e na manutenção em patamares mínimos dos custos operacionais. Essa realidade afeta tanto as empresas de origem privada, quanto aquelas de origem pública, e em especial as que necessitam de um sistema logístico que dê suporte as suas demandas por recursos materiais e humanos.

De um modo geral, muito tem se estudado e discutido sobre a logística de abastecimento (*in-bound logistics*) e a logística de distribuição (*out-bound logistics*). Porém, é preciso considerar também o papel da logística interna a organização, através da qual é feita a alocação dos recursos necessários à produção de bens ou serviços. Muitos dos conceitos e técnicas empregados no planejamento e operação dos canais de abastecimento e distribuição podem ser aplicados na logística interna, tendo como objetivo a minimização dos custos operacionais e do tempo de resposta das solicitações que ocorrem entre os vários estágios do processo produtivo.

Neste enfoque da logística interna, é possível notar que em um número significativo de empresas a própria natureza do processo produtivo exige a movimentação constante de recursos, seja em âmbito interno ou externo aos seus limites físicos. Pode-se identificar assim, a atividade de transporte como um elemento de fundamental importância dentro do processo logístico.

Outro fato que ressalta a importância a ser dada a atividade de transporte está relacionado ao impacto de seu custo sobre os custos logísticos totais, cujo percentual de participação pode variar de 30% a 60% (BALLOU, 1999).

Deste modo, a necessidade de uma busca permanente pela eficiência dos processos que compõem a atividade de transporte abre uma importante frente para estudos e pesquisas que possam resultar em benefícios para as empresas e para os usuários. Temas a serem tratados nesta

área envolvem o gerenciamento de frotas, de custos e de risco; a rastreabilidade; além de um outro que se destaca pelo elevado grau de complexidade matemática e computacional: a roteirização de veículos.

O termo *roteirização* pode ser definido como a determinação da melhor seqüência em que vias e/ou pontos devem ser percorridos por veículos, visando o atendimento das demandas por serviço e tendo como objetivo minimizar os custos operacionais, as distâncias percorridas ou os tempos dos trajetos. Decisões a serem tomadas neste tipo de problema se referem à definição dos melhores trajetos que devem ser executados através de malhas rodoviárias, de redes viárias urbanas, de linhas ferroviárias, e de linhas ou rotas de navegação aquaviária ou aérea.

Apesar de serem mais conhecidos os casos que envolvem a roteirização de veículos, a definição dos trajetos que deverão ser percorridos por pessoas que executam algum tipo de serviço ao longo de um sistema viário se enquadra no mesmo tipo de situação. Desta forma, os princípios utilizados na modelagem dos problemas de roteirização de veículos são aplicáveis também ao caso da roteirização de pessoas.

Embora os problemas de roteirização apresentem variações, é possível reduzi-los, de um modo geral, segundo as características da origem e destino do trajeto e segundo o tipo de modelagem.

Quanto à origem e destino, há o problema de se encontrar um trajeto em uma rede de transporte onde o ponto de origem seja diferente do ponto de destino. Existem ainda os casos em que os pontos de origem e de destino são coincidentes e os casos em que existem múltiplos pontos de origem e de destino.

No que diz respeito ao tipo de modelagem, os problemas de roteirização apresentam duas classes básicas:

a) Problemas de Roteirização em Nós (*Node Routing Problems*), nos quais os locais de atendimento são representados como pontos específicos em uma rede viária, caracterizados como nós ou vértices;

b) Problemas de Roteirização em Arcos (*Arc Routing Problems*), nos quais os locais de atendimento são representados de forma contínua ao longo dos segmentos de via, caracterizados como arcos e arestas.

A generalização destes dois casos em um único problema é conhecida como Problema de Roteirização Geral (*General Routing Problem*).

O Problema de Roteirização em Nós consiste em determinar, por exemplo, qual a melhor seqüência em que os pontos de demanda dispersos em uma certa região devem ser visitados por um veículo que parte de um depósito central, realiza tarefas de coleta e/ou entrega e retorna ao final para o ponto de origem. Podem ser consideradas neste caso ainda, restrições como o limite de carregamento do veículo, o tempo máximo de jornada de trabalho da tripulação e as faixas horárias em que os clientes devem ser atendidos que também são conhecidas como *janelas de tempo*.

Já no caso dos Problemas de Roteirização em Arcos o objetivo é determinar um percurso de custo mínimo através dos segmentos de via de uma rede de transporte. Este tipo de problema aparece nos serviços de varrição de ruas por equipamentos mecânicos; de remoção de neve das vias; de aspersão de sal em vias como medida preventiva ao congelamento; de coleta de lixo; de entrega de correspondência, de jornais e de folhetos de publicidade; de leitura de medidores de energia elétrica, de água e de gás; de patrulhamento de ruas por viaturas policiais; de fixação de componentes eletrônicos em placas de circuitos; de inspeção de redes elétricas; e no transporte escolar por ônibus.

Nos exemplos acima mencionados, nota-se que uma parcela significativa dos serviços modelados como Problema de Roteirização em Arcos são de caráter essencial e apresentam utilização intensiva de mão-de-obra e equipamentos. É possível, então, perceber o impacto

negativo que um planejamento inadequado destas atividades, no tocante a utilização dos recursos disponíveis, pode ter sobre os custos operacionais e sobre o nível de serviço prestado aos clientes.

Por outro lado, fatos como a mudança no comportamento dos consumidores, a regulação de serviços públicos concedidos e a popularização de novas tecnologias especialmente no campo da telemática, geram a necessidade de um contínuo aperfeiçoamento dos procedimentos operacionais adotados pelas empresas. Em termos de modelagem matemática isto implica na consideração de restrições adicionais que elevam significativamente o grau de complexidade do problema de roteirização.

Além das questões relacionadas à complexidade matemática, pode ser notada também a importância dada, pelos meios empresariais e acadêmicos, da presença de um modelo de roteirização no ambiente gerencial como parte integrante de um sistema amplo de apoio à tomada de decisão e que seja capaz de levar em conta os aspectos fundamentais das operações realizadas. Este fato pode ser verificado pelo crescimento significativo da oferta no mercado de softwares de pacotes aplicativos com rotinas específicas para roteirização.

Entretanto, deve-se considerar que, apesar da popularização e queda nos custos de aquisição destes pacotes, sua aplicação se limita a propósitos gerais. Isto se deve naturalmente, a dificuldade de customização face às particularidades apresentadas pelo processo operacional de cada empresa.

Tendo como base seus objetivos e as oportunidades de desenvolvimento advindas da importância dos modelos de roteirização no ambiente gerencial, será tratado neste trabalho o caso específico do Problema de Roteirização em Arcos.

1.2 OBJETIVOS

Os objetivos deste trabalho estão divididos em dois níveis. O primeiro deles compreende o *objetivo geral*. No segundo nível aparecem os *objetivos específicos*.

1.2.1 Objetivo Geral

A presente dissertação de mestrado tem com objetivo principal a proposição de um modelo de resolução para o Problema de Roteirização em Arcos Capacitado - PRAC, aplicável no caso de redes de transporte representadas através de grafos não orientados.

1.2.2 Objetivos Específicos

Os objetivos específicos desta dissertação de mestrado são:

a) Desenvolver um procedimento para construção de partições, que são os conjuntos de segmentos de via que cada veículo ou pessoa deve atender;

b) Desenvolver um procedimento para melhoria da solução inicial através da troca de segmentos entre partições adjacentes;

c) Definir os algoritmos de roteirização que devem ser agregados ao modelo.

1.3 JUSTIFICATIVAS

Os modelos de roteirização, de um modo geral, representam uma importante área de pesquisa em logística, uma vez que os algoritmos de solução exigem um contínuo aperfeiçoamento visando agregar elementos que reflitam melhor as operações reais (DASKIN, 1985).

Com base nesta necessidade, a produção acadêmica sobre os Problemas de Roteirização em Nós tem sido bastante numerosa nos últimos 45 anos. Embora mais restrito, o desenvolvimento de modelos de resolução na área dos Problemas de Roteirização em Arcos não é menos importante, principalmente pelo seu amplo campo de aplicação.

No caso específico dos Problemas de Roteirização em Arcos, podem ser notadas ainda novas oportunidades decorrentes do pequeno número de trabalhos publicados nos quais é dada ênfase à proposição de modelos para resolução das versões capacitadas.

Além disso, poucos trabalhos entre aqueles dedicados ao Problema de Roteirização em Arcos Capacitado consideram a estratégia de *agrupar primeiro – roteirizar depois*, passível de ser aplicada nas situações em que é necessária uma definição clara da área de atendimento de cada veículo ou pessoa em uma rede de transporte.

1.4 HIPÓTESE PARA REALIZAÇÃO DO ESTUDO

A proposição do modelo para resolução do Problema de Roteirização em Arcos Capacitado baseia-se na seguinte hipótese:

> *A aplicação de um modelo, desenvolvido a partir de princípios matemáticos, na resolução de um problema combinatorial de elevado grau de complexidade como é o Problema de Roteirização em Arcos Capacitado, possibilita a obtenção de melhores soluções do que aquelas geradas a partir do emprego de métodos empíricos baseados unicamente na experiência prática de um planejador.*

1.5 MÉTODO DE ESTUDO

Para que pudessem ser atingidos os objetivos propostos nesta dissertação de mestrado foi adotado um método de estudo composto das seguintes etapas:

a) Definição, elaboração e apresentação de um projeto de dissertação;

b) Revisão da literatura compreendendo os aspectos conceituais e resolutivos dos Problemas de Roteirização em Arcos e do Problema de Particionamento de Redes de Transporte;

c) Desenvolvimento do modelo compreendendo a elaboração de um algoritmo para resolução do Problema de Particionamento de Redes de Transporte representadas por grafos não orientados; a definição dos algoritmos de roteirização a serem empregados no modelo; e a implementação computacional do modelo;

d) Aplicação do modelo proposto através da realização de um estudo de caso, seguindo-se a análise dos resultados obtidos e a comparação com a prática atualmente utilizada;

e) Elaboração das conclusões e da proposta para futuros desenvolvimentos.

1.6 DELIMITAÇÕES DO TRABALHO

O trabalho desenvolvido nesta dissertação de mestrado apresenta as seguintes delimitações:

a) O modelo proposto é aplicável somente no caso de redes de transporte representadas por grafos não orientados;

b) Dada a abordagem heurística, segundo a qual foi desenvolvido o modelo, são obtidas boas soluções (que não são ótimas) em um tempo de processamento computacional aceitável;

c) As demandas por serviço existentes nos segmentos de via são determinísticas;

d) A restrição considerada no modelo contempla o tempo máximo da jornada de trabalho das tripulações de veículos ou membros de uma equipe de trabalho.

1.7 ESTRUTURA DO TRABALHO

Além do presente capítulo, de caráter introdutório, este trabalho é composto por mais 5 capítulos.

O capítulo 2 apresenta uma revisão bibliográfica dos Problemas de Roteirização em Arcos. Já no capítulo 3 é apresentada uma revisão bibliográfica do Problema do Particionamento de Redes de Transporte. Nestes dois capítulos é dada ênfase aos aspectos conceituais e resolutivos.

No capítulo 4 é feita a proposição de um modelo para a resolução do Problema de Roteirização em Arcos Capacitado no caso de redes de transporte representadas por grafos não orientados. Neste capítulo é descrita também a implementação computacional do modelo.

O capítulo 5 apresenta a aplicação do modelo proposto em um estudo de caso, e a seguir, uma análise dos resultados obtidos com esta aplicação e a sua comparação com a prática atualmente adotada pela empresa executora do serviço. Por último, no capítulo 6 são apresentadas as conclusões deste trabalho e as propostas para futuros desenvolvimentos.

2 PROBLEMAS DE ROTEIRIZAÇÃO EM ARCOS

2.1 INTRODUÇÃO

Os Problemas de Roteirização em Arcos, assim como qualquer outro modelo matemático de cunho aplicativo, tem seus processos de formulação e resolução alicerçados sobre um conjunto de princípios teóricos. Deste modo, para que seja obtido um perfeito entendimento do problema a ser tratado, é necessária uma análise prévia de tais princípios.

Tendo em vista esta necessidade, neste capítulo são apresentados os aspectos conceituais e resolutivos dos Problemas de Roteirização em Arcos. No item 2.2 é feita uma descrição do histórico de seu desenvolvimento. O item 2.3 apresenta os aspectos conceituais dos Problemas de Roteirização em Arcos, no tocante as estruturas de representação e aos tipos de percurso que são definidos em uma rede de transporte. A subclassificação adotada nos Problemas de Roteirização em Arcos é apresentada no item 2.4. No item 2.5 é feita referência as variantes dos Problemas de Roteirização em Arcos. O item 2.6 apresenta o Problema de Roteirização em Arcos Capacitado, em cujo modelo são consideradas restrições de ordem operacional. Por último, no item 2.7, são feitas as considerações finais deste capítulo.

2.2 HISTÓRICO DO DESENVOLVIMENTO DOS PROBLEMAS DE ROTEIRIZAÇÃO EM ARCOS

A primeira referência sobre um Problema de Roteirização em Arcos se confunde com a proposição da *Teoria dos Grafos*, feita pelo matemático suiço Leonhard Euler, em 1736, a partir do caso das pontes de Konigsberg. Tal problema consistia em definir um percurso para uma procissão que, partindo de um determinado ponto, atravessasse exatamente uma vez cada uma das sete pontes sobre o Rio Pregel (figura 1) e se encerrasse no mesmo ponto de partida. Euler demonstrou a impossibilidade da definição de um percurso com tais características neste caso em particular, porém estabeleceu as condições necessárias para a sua existência. Surge daí a expressão *ciclo de Euler*.

Passados dois séculos desde a publicação do trabalho de Euler, o advento da Segunda Guerra Mundial trouxe consigo a necessidade de modelos quantitativos que fossem capazes de auxiliar a tomada de decisão sobre a alocação de recursos em sistemas logísticos bastante complexos. Com base nesta necessidade foram desenvolvidos os primeiros modelos de Pesquisa Operacional em meados da década de 40.

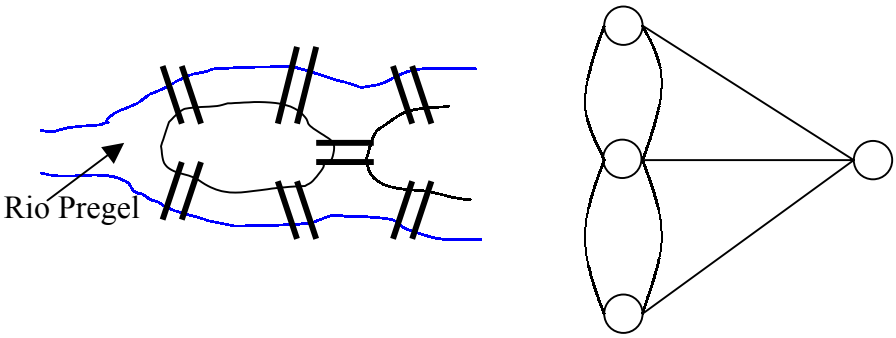


Figura 1 - Pontes de Königsberg e o multigrafo correspondente

Com o término da Segunda Guerra Mundial o emprego dos modelos de Pesquisa Operacional deixou de ser restrito ao âmbito militar e se estendeu ao ambiente gerencial em organizações públicas e privadas, bem como ao ensino e pesquisa nas universidades. Nestas últimas em particular, a Pesquisa Operacional se tornou uma importante disciplina em áreas do conhecimento que lidam com o gerenciamento de recursos materiais, financeiros e humanos destinados a produção de bens e serviços. Ao mesmo tempo, o interesse despertado em

pesquisadores passou a motivar um intenso desenvolvimento de novas técnicas aplicáveis a este campo do conhecimento.

A Pesquisa Operacional engloba várias técnicas de programação matemática como a programação linear, a programação não linear e a programação inteira. É importante ressaltar que o termo *programação* deve ser entendido aqui de maneira diferente daquela usada em computação na qual está relacionada à implementação de um algoritmo qualquer em uma determinada linguagem de programação. Programação aqui é usada no sentido de planejamento, ou seja, o planejamento matemático de um sistema em estudo (GOLDBARG e LUNA, 2000).

Á partir da metade da década de 1950 teve início um intenso desenvolvimento de novas famílias de modelos de otimização que representaram um importante avanço para a resolução dos Problemas de Roteirização em Arcos: os *modelos de conexão* e os *modelos de fluxos em redes*. Nos quadros 1 e 2 são citados, respectivamente, os modelos de conexão e os modelos de fluxos em redes e os principais algoritmos de solução. Informações mais detalhadas sobre estes modelos e seus algoritmos de solução podem ser obtidas nos trabalhos de Syslo *et al* (1983), Gondran e Minoux (1985) e Prins (1994).

Quadro 1 - Modelos de conexão e algoritmos de solução

Modelo de Conexão	**Algoritmo de Solução**
Caminho Mínimo	Dijkstra Ford-Moore-Bellman Floyd-Warshall
Árvore de Peso Mínimo	Prim Kruskal
Emparelhamento	Edmonds Gabow Micali e Vazirani

Quadro 2 - Modelos de fluxos em redes e algoritmos de solução

Modelo de Fluxo em Redes	**Algoritmo de Solução**
Fluxo Máximo	Ford-Fulkerson
Fluxo de Custo Mínimo	Busacker

Em 1962 é publicado pelo matemático chinês Meiko Guan um dos mais celebres trabalhos tratando do Problema de Roteirização em Arcos. Guan, que durante a Revolução Cultural Chinesa havia trabalhado por um período de tempo em uma agência dos correios, levantou o problema de se determinar um percurso de extensão mínima para um carteiro que, partindo de uma agência deve atravessar pelo menos uma vez cada segmento de via em uma área de atendimento, concluindo seu trajeto no ponto de partida. Esta é a origem do *Problema do Carteiro Chinês* – PCC (EISELT *et al*, 1995a).

A década de 1970 é marcada por um grande avanço no desenvolvimento de modelos matemáticos destinados à resolução dos Problemas de Roteirização em Arcos. Edmonds e Johnson (1973) publicaram um trabalho, considerado básico até os dias atuais, no qual foram estabelecidos os principais algoritmos de solução do PCC em redes representadas por grafos não orientados, orientados e mistos.

A seguir, Orloff (1974), propõe o *Problema do Carteiro Rural* - PCR, cuja diferença em relação ao PCC reside no fato de nem todos segmentos de uma rede de atendimento terem demanda por serviço, o que não obriga seu percurso. A denominação adotada para este problema é derivada da semelhança existente com o tipo de percurso realizado por um carteiro em uma zona rural.

Ainda no mesmo ano, Beltrami e Bodin (1974) publicam um importante trabalho relacionando os aspectos conceituais dos Problemas de Roteirização em Arcos com as aplicações de ordem prática, mais especificamente no serviço de coleta de resíduos na área urbana de Nova York. Outro trabalho de destaque relacionado a uma aplicação prática foi publicado por Stern e Dror (1979) envolvendo a roteirização de leituristas de medidores de energia elétrica na cidade de Beershava, em Israel.

No final da década de 1970, dois trabalhos publicados por Bodin e Kursh (1978, 1979) relatam uma das primeiras experiências de desenvolvimento e implantação de um sistema para planejamento dos roteiros de equipamentos utilizados na varrição de ruas em Nova York. Nestes dois trabalhos já podia ser identificada uma nova tendência que veio a se consolidar na década

seguinte: a consideração dos problemas de roteirização como parte integrante de um sistema abrangente de apoio à tomada de decisão gerencial.

Em meados da década de 1980 foram lançados no mercado os primeiros aplicativos computacionais capazes de armazenar, manipular e apresentar sob a forma gráfica informações contidas em bases de dados georeferenciadas, os *Sistemas de Informações Geográficas* - SIG. Esta nova tecnologia rapidamente encontrou um vasto campo de aplicação junto aos problemas de roteirização dada à necessidade de se considerar informações referentes a sistemas viários bastante complexos, além da localização e quantificação das demandas por serviço. Um dos primeiros trabalhos descrevendo a aplicação de um SIG no processo de resolução de um Problema de Roteirização em Arcos foi publicado por Bodin *et al* (1989), tendo sido tratado o caso da coleta de resíduos sólidos no distrito de Oyster Bay em Nova York.

O desenvolvimento de recursos de computação gráfica possibilitou também a implementação de uma interface mais amigável entre os usuários e os sistemas computacionais que executam a resolução dos modelos de roteirização. Desta forma, puderam ser adotadas novas abordagens capazes de combinar o uso de algoritmos de solução teóricos com a intervenção do usuário no sentido de buscar a melhoria de uma solução inicial a partir do seu julgamento e da habilidade pessoal. Exemplo disso foi à abordagem desenvolvida por Li e Eglese (1996) para o caso da roteirização de equipamentos de aspersão de sal no condado de Lancashire.

Atualmente, tem sido notável o avanço técnico dos recursos computacionais, principalmente em relação à velocidade de processamento e a capacidade de armazenamento de dados até mesmo em micro-computadores pessoais, além da redução dos custos de aquisição e desenvolvimento de sistemas. Com isso, novas e importantes perspectivas de desenvolvimento estão sendo abertas na área dos problemas de roteirização, sejam eles modelados como nós ou como arcos.

A primeira grande oportunidade de desenvolvimento que pode ser citada está relacionada com a possibilidade do tratamento de problemas cada vez mais complexos, nos quais são bastante significativos a quantidade de variáveis de decisão e o grau de dificuldade das restrições a serem

consideradas. Nestas situações se torna imprescindível o emprego de técnicas mais avançadas de programação matemática, as quais exigem o uso de recursos computacionais mais sofisticados. Desenvolvimentos desta ordem possibilitam a obtenção de resultados finais de boa qualidade, o que no caso de problemas de natureza combinatorial como são os de roteirização significa valores cada vez mais próximos ao ótimo.

Um outro campo que apresenta boas perspectivas de desenvolvimento é aquele que vem sendo denominado, em alguns trabalhos científicos na área de roteirização (KEENAN, 1998; TARANTILIS e KIRANOUDIS, 2002), como *Sistemas de Apoio a Tomada de Decisão Espacial* (do inglês, *Spatial Decision Support Systems – SDSS*). Nestes sistemas os modelos de roteirização são incorporados a ferramentas gerenciais mais amplas que possibilitam a comunicação e a troca de dados entre os vários setores de uma organização privilegiando a utilização de informações georeferenciadas.

Não podem deixar de serem citadas ainda novas tendências tecnológicas consideradas em logística como o *e-commerce*, as demandas estocásticas e a rastreabilidade. Tais inovações afetam significativamente o nível de serviço e, conseqüentemente a competitividade das organizações, o que acaba por exigir sistemas de roteirização cada vez mais sofisticados e que sejam capazes de atender as necessidades específicas dos ambientes nos quais são implantados (PARTYKA e HALL, 2002).

2.3 ASPECTOS CONCEITUAIS DOS PROBLEMAS DE ROTEIRIZAÇÃO EM ARCOS

Seja $G(V, A \cup E)$ um grafo que representa uma determinada rede viária, onde $V = \{v_i, ..., v_n\}$é um conjunto finito de *vértices* (também chamados de *nós*) ou interseções de vias; $A = \{a_{ij},$ onde $i \neq j\}$ é um conjunto finito de *arcos* ou segmentos de via com um sentido de percurso obrigatório; e $E = \{e_{ij},$ onde $i < j\}$ é um conjunto finito de *arestas* ou segmentos de via sem um sentido de percurso obrigatório. A cada arco ou aresta de G pode estar associado um coeficiente de custo c_{ij}, não negativo ou ∞ se este não for definido, que pode representar o comprimento do segmento, o tempo de percurso do segmento ou uma função de custo generalizado, tornando assim, G um grafo valorado. Quando dois ou mais segmentos incidem em dois vértices

adjacentes, *G* é denominado como *multigrafo* (figura 2). Um grafo é *conexo* se todos os vértices estiverem ligados por arcos ou arestas.

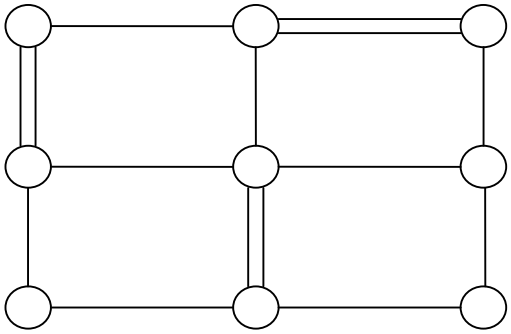


Figura 2 – Representação de um multigrafo

Um grafo pode ser definido como:

a) *orientado*: quando todos segmentos possuem um sentido obrigatório de percurso, sendo representado por $G\ (V, A)$ e onde $E = \phi$;

b) *não orientado*: quando todos os segmentos não possuem um sentido de percurso obrigatório, sendo representado por $G\ (V, E)$ e onde $A = \phi$;

c) *misto*: se existem segmentos com um sentido obrigatório de percurso e também segmentos sem um sentido obrigatório de percurso, sendo representado por $G\ (V, A \cup E)$.

Um *ciclo simples* é um percurso no qual cada segmento de *G* é atravessado no máximo uma vez. Já em um *ciclo de Euler* cada segmento de *G* é atravessado exatamente uma vez. Um *ciclo de carteiro* em *G* é um percurso que atravessa cada segmento pelo menos uma vez, sendo este o caso estabelecido no PCC.

Um *caminho* é um percurso semelhante a um ciclo, exceto pelo fato de que os pontos de início e fim não são coincidentes. Se em *G* existirem dois vértices de grau ímpar, dados por *s* e *t*,

sendo os demais vértices de grau par, pode ser definido um percurso que inicia em *s*, atravessa exatamente uma vez cada segmento e termina em *t*.

Um ciclo ou um caminho pode ser representado por uma seqüência do tipo v_1, e_{12}, v_2, e_{2n}, v_n de vértices e arestas, para o caso dos grafos não orientados. Vale o mesmo principio para os grafos orientados e mistos devendo ser considerados, porém, os respectivos elementos que são os arcos.

Se estiver associado a cada segmento de *G* um coeficiente de custo c_{ij}, então o ciclo ou o caminho terá um custo total (C_T) dado por:

$$C_T = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} \qquad (1)$$

Quando um grafo conexo tiver todos os seus vértices com grau par, ou seja, incididos por um número par de segmentos, será denominado como *grafo euleriano*. Se tais condições forem satisfeitas será possível determinar um ciclo de Euler em *G*.

Por outro lado, quando o grafo não é euleriano, isto é, possuir também vértices de grau ímpar (incididos por um número ímpar de segmentos), o ciclo a ser aí construído compreenderá também segmentos que serão percorridos mais de uma vez e que representam os trechos de percurso ocioso. Sempre que existir um ciclo de Euler em *G*, este irá representar a solução ótima para o PCC.

Para maiores informações sobre os aspectos conceituais dos Problemas de Roteirização em Arcos podem ser consultados os trabalhos de Gondran e Minoux (1985), Eiselt *et al* (1995a; 1995b) e Araújo e Michel (2001).

2.4 SUBCLASSIFICAÇÃO DOS PROBLEMAS DE ROTEIRIZAÇÃO EM ARCOS

Seja $A \cup E$ o conjunto de todos os segmentos existentes em G e $A_R \subseteq A \cup E_R \subseteq E$ o subconjunto dos segmentos de G que apresentam demanda por serviço. Assim, os Problemas de Roteirização em Arcos são divididos nas duas seguintes subclasses básicas:

a) *Problema do Carteiro Chinês* – PCC, quando todos os segmentos de G apresentam demanda por serviço ($A_R = A \cup E_R = E$) e devem ser percorridos pelo menos uma vez (EDMONDS e JOHNSON, 1973);

b) *Problema do Carteiro Rural* – PCR, quando alguns dos segmentos de G apresentam demanda por serviço ($A_R \subset A \cup E_R \subset E$) e devem percorridos pelo menos uma vez (ORLOFF, 1974).

A descrição destas subclasses dos Problemas de Roteirização em Arcos é apresentada nos dois subitens a seguir.

2.4.1 O Problema do Carteiro Chinês

O Problema do Carteiro Chinês (PCC) pode ser definido em grafos não orientados, orientados e mistos.

No caso não orientado, o problema mais simples de ser tratado ocorre quando todos os vértices de G são de grau par, sendo que o ciclo de Euler aí existente representa a solução ótima. O problema resume-se então a determinação da seqüência em que os segmentos de G devem ser percorridos.

Edmonds e Johnson (1973) apresentaram três algoritmos, denominados respectivamente *End-pairing*, *Next-node* e *Maze-search* para a determinação de um ciclo de Euler em um grafo não orientado onde todos os vértices são de grau par. O *End-pairing* que possui complexidade da ordem de $O(|V|)$ é descrito abaixo:

Passo 1. Trace um ciclo simples que pode não conter todos os vértices de *G*. Se todas as arestas foram incluídas no ciclo, então pare.

Passo 2. Considere um vértice qualquer v_i no ciclo incidido por uma aresta ainda não incluída no ciclo. Construa um segundo ciclo a partir de v_i não sobrepondo o primeiro.

Passo 3. Sejam e_1, e_2 as duas arestas incidentes em v_i no primeiro ciclo e e_3, e_4 as arestas incidentes em v_i no segundo ciclo. Junte os dois ciclos em um único. Se todas as arestas de *G* foram percorridas então pare. Senão retorne ao passo 2.

A figura 3 mostra um grafo euleriano e conexo no qual um ciclo de Euler é dado pela seqüência dos vértices 1, 2, 5, 3, 4, 5, 7, 6, 4, 1.

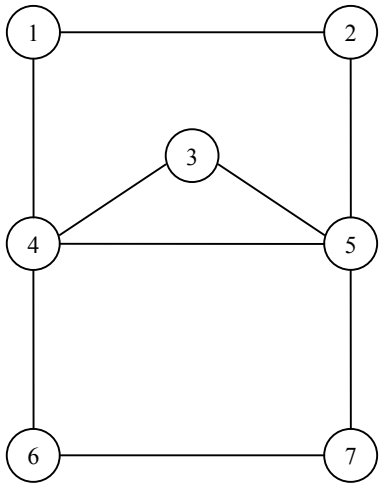


Figura 3 - Grafo não orientado com ciclo de Euler associado

O PCC não orientado pode ser formulado da seguinte maneira:

$$\text{Minimizar } z = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \qquad (2)$$

Sujeito a:

$$\sum_{j=1}^{n} x_{ji} - \sum_{j=1}^{n} x_{ij} = 0 \qquad i = 1, ..., n. \qquad (3)$$

$$x_{ij} + x_{ji} \geq 1 \qquad \forall\ (i, j) \in E \qquad (4)$$

$$x_{ij} \geq 0 \text{ e inteiro} \qquad (5)$$

onde:

x_{ij} é o número de vezes que a aresta e_{ij} é percorrida de i para j;
c_{ij} é o coeficiente de custo associado a aresta e_{ij}.

No modelo matemático apresentado acima, a restrição (3) garante a continuidade do percurso, a restrição (4) que nenhuma aresta deixará de ser atravessada e a restrição (5) a não-negatividade e integralidade da solução a ser obtida.

Quando um grafo não orientado e conexo possuir vértices de grau ímpar, a solução do PCC será obtida através da transformação de G em um grafo aumentado G' no qual todos os vértices terão grau par. Será possível determinar, então, um ciclo de Euler cujo percurso ocioso seja mínimo. O grafo G' é obtido a partir da adição em G de cópias de algumas arestas de E a mínimo custo. Uma vez que todo grafo conexo possui sempre um número par de vértices de grau ímpar, a determinação de G' pode ser feita com a adição de arestas em G que liguem justamente os vértices de grau ímpar (EISELT *et al*, 1995a).

Se g_i é o grau de um vértice v_i e m o número de arestas em E, então $\sum_i g_i = \sum_{i \in par} g_i + \sum_{i \in ímpar} g_i = 2m$ porque cada aresta adiciona uma unidade ao grau de cada um de seus vértices extremos. Uma vez que $\sum_{i \in par} g_i$ é par, o $\sum_{i \in ímpar} g_i$ também será par. Seja V_o o conjunto dos vértices de grau ímpar, V_e o conjunto dos vértices de grau par e V o conjunto de todos os vértices de G. Como o número de vértices de grau ímpar (V_o) é par, estes vértices podem ser divididos em dois grupos e formar então $r = (1/2)\,|V_o|$ caminhos entre pares de vértices que estejam em grupos diferentes. As arestas E' contidas nestes caminhos são adicionadas ao grafo original como as arestas artificiais necessárias para a formação de G'. O grafo G' poderá se tornar um multigrafo, já que as arestas artificiais são cópias das arestas originais de G. Uma solução viável para o PCC se origina, portanto, dos r caminhos que conectam os r pares vértices de grau

ímpar. O problema se reduz assim à determinação dos melhores *r* caminhos, que representam o mínimo custo (BODIN *et al*, 1983).

Um modo de formular o PCC em um grafo conexo não orientado como um problema de programação linear inteira pode ser feito quando se busca a construção de *G'*. Para isto, a variável x_{ij} é definida como o número de cópias de e_{ij} ($i < j$) necessárias para aumentar *G*. Sejam também δ_i o conjunto das arestas incidentes no vértice v_i e $V_o \subseteq V$ o conjunto dos vértices de grau ímpar em *V*. A formulação pode ser dada então por:

$$\text{Minimizar } z = \sum_{e_{ij} \in E} c_{ij} x_{ij} \tag{6}$$

Sujeito a:

$$\sum_{e_{ij} \in E} x_{ij} \equiv \begin{cases} 1 \pmod 2 \text{ se } v_i \in V_o \\ \\ \\ 0 \pmod 2 \text{ se } v_i \in V \setminus V_o \end{cases} \tag{7}$$

$$x_{ij} \in \{0, 1\}, e_{ij} \in E \tag{8}$$

Edmonds e Johnson (1973) demonstraram que a definição dos *r* melhores caminhos entre os vértices de grau ímpar pode ser feita a partir da resolução de um problema de emparelhamento perfeito de peso mínimo. Seja $R_T = (V_0, F')$ um grafo completo formado a partir do grafo original *G*, onde $F' = \{(v_i, v_j): v_i, v_j \in V_o, i < j\}$ e o coeficiente de custo de cada elemento de *F'* corresponde ao caminho mínimo entre v_i e v_j. O grafo *G'* é obtido, então, a partir da adição a *G* dos caminhos mínimos que correspondem à solução ótima do problema de emparelhamento perfeito.

A solução deste problema em tempo polinomial ($O|V|^3$) foi proposta por Edmonds e Johnson (1973) através da adaptação do *Blossom Algorithm* desenvolvido anteriormente por Johnson para a resolução do problema da determinação de um emparelhamento máximo. Para maiores detalhes sobre esta abordagem podem ser consultados os trabalhos de Edmonds e

Johnson (1973) e Gondran e Minuox (1985). Algoritmos com complexidade igual ou inferior a $O(|V|^3)$ para a determinação de emparelhamentos perfeitos de peso mínimo são apresentados nos trabalhos de Gabow (1976), Syslo *et al* (1983), Galil *et al* (1986), Gabow e Tarjan (1991) e Derigs e Metz (1991).

Assim, o algoritmo de solução do PCC não orientado pode ser dado pela seqüência abaixo:

Passo 1. Leia o grafo *G* (*V, E*). Se todos os vértices de *G* tem grau par vá para o passo 4, senão vá para o passo 2.

Passo 2. Reúna todos os vértices de grau ímpar de *G* e construa um grafo completo R_T associando a cada par de vértices v_i e v_j de R_T uma aresta com peso igual ao caminho mais curto que liga v_i a v_j em *G*.

Passo 3. Determine o emparelhamento perfeito de peso mínimo *M'* em R_T. Para cada aresta de *M'* associe uma nova aresta em *G* no caminho mínimo que ela representa, obtendo um grafo *G'*.

Passo 4. Determine a solução do PCC que é representada por um ciclo de Euler em *G'*.

Como exemplo de solução do PCC não orientado, seja o grafo mostrado na figura 4. Deseja-se determinar um ciclo de extensão mínima com início e fim no vértice 1. Os coeficientes de custo das arestas representam as distâncias (em unidades de comprimento – u.c.) entre dois vértices adjacentes.

A existência de vértices de grau impar (2, 3, 5, 8, 10 e 11) torna necessária a construção de um grafo aumentado *G'*. Como $V_o = 6$, deverão existir $r = (1/2)|6| = 3$ caminhos que representam o emparelhamento perfeito de peso mínimo no grafo completo R_T.

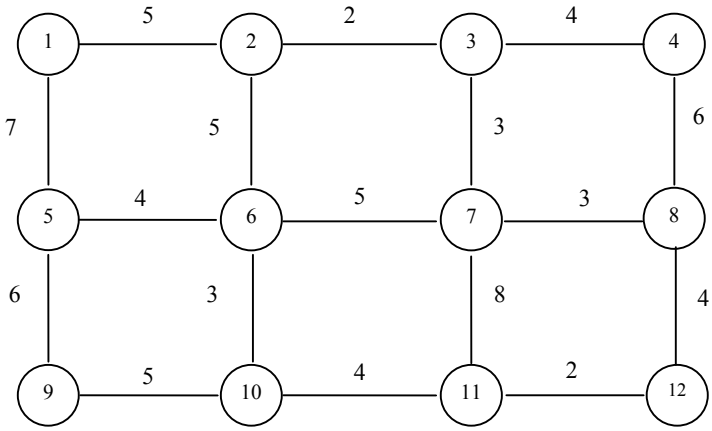


Figura 4 - Grafo não euleriano valorado

O grafo completo R_T formado pelos elementos de V_o e pelas arestas que representam os caminhos mínimos em G é mostrado na figura 5.

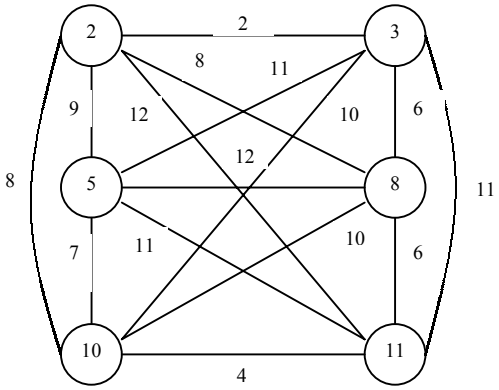


Figura 5 - Grafo completo formado a partir dos elementos de V_o e E'

A solução ótima para o emparelhamento perfeito de peso mínimo obtido em R_T é dada pelas arestas compreendidas entre os pares de vértices (2, 3), (5, 10) e (8, 11). A figura 6 mostra o grafo aumentado G', que é formado a partir da replicação de arestas incidentes nos vértices de grau ímpar (representadas pelas linhas tracejadas) e que serão atravessadas mais de uma vez (percurso ocioso). Pode ser definido, então, um ciclo de carteiro dado pela seqüência de vértices 1, 2, 3, 2, 6, 7, 3, 4, 8, 12, 8, 7, 11, 12, 11, 10, 6, 10, 9, 5, 6, 5, 1 com um custo total de 91 u. c.

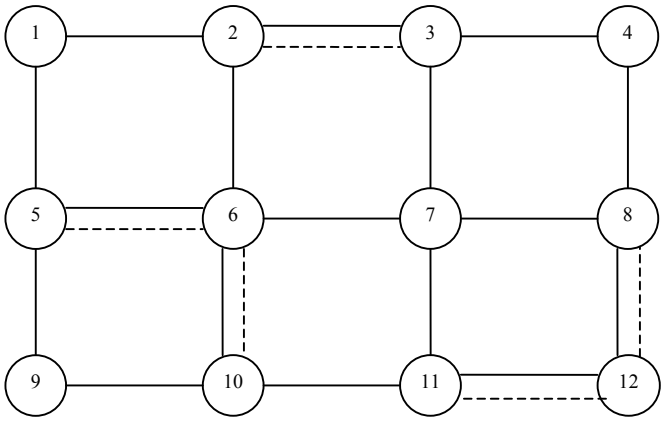


Figura 6 - Grafo aumentado

No caso dos grafos orientados conexos, a condição suficiente para a existência de um ciclo de Euler é a de que cada vértice tenha semigrau nulo, ou seja, a diferença entre o número de arcos chegando ao vértice (semigrau interior) e o número de arcos partindo do vértice (semigrau exterior) seja igual a zero (BODIN *et al* 1983; GOLDBARG e LUNA, 2000). Ocorrendo esta situação o grau de todos vértices será par.

Entretanto, além da condição de todos os vértices terem semigrau nulo, é necessário que *G* seja um grafo *fortemente conexo*, isto é, um grafo no qual possam ser determinados caminhos ligando cada vértice aos demais (figura 7).

Quando um grafo orientado fortemente conexo possuir vértices com semigrau diferente de zero, será necessária a sua transformação em um grafo aumentado *G'*, no qual todos os vértices tenham semigrau nulo. A obtenção de *G'* pode ser feita, em tempo polinomial, através da resolução de um problema de fluxo de custo mínimo no qual o fluxo em cada arco seja no mínimo igual a 1 (EDMONDS e JOHNSON, 1973; ORLOFF, 1974; BELTRAMI e BODIN, 1974).

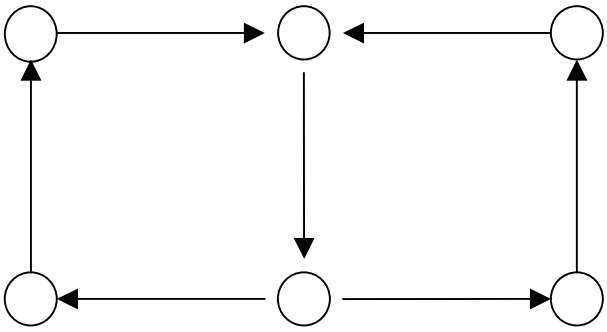


Figura 7 - Grafo fortemente conexo

Seja I um conjunto de vértices v_i de G nos quais o número de arcos chegando excede o número de arcos partindo por um valor s_i; e J um conjunto de vértices v_j de G nos quais o número de arcos partindo excede o número de arcos chegando por um valor d_j. As variáveis s_i e d_j podem ser interpretadas respectivamente como pontos de fornecimento e demanda, onde $\sum_{n_i \in I} s_i = \sum_{n_j \in J} d_j$.

Com isto, é construído, a partir de G, um grafo bipartido R_B com os vértices contidos em I e J. O coeficiente de custo c_{ij} de cada arco de R_B representa o caminho mínimo entre v_i e v_j em G.

Assim, para que um ciclo de Euler possa ser determinado em G', cada vértice v_i deve ter um número adicional de arcos partindo e cada vértice v_j deve ter um número adicional de arcos chegando. O PCC em um grafo orientado fortemente conexo pode ser formulado como:

$$\text{Minimizar } z = \sum_{v_i \in I} \sum_{v_j \in J} c_{ij} x_{ij} \qquad (9)$$

Sujeito a:

$$\sum_{v_j \in J} x_{ij} = s_i \quad (v_i \in I) \qquad (10)$$

$$\sum_{v_i \in I} x_{ij} = d_j \quad (v_j \in J) \qquad (11)$$

$$x_{ij} \geq 0 \qquad (12)$$

O valor ótimo de x_{ij} resultante representa o número extra de vezes que cada arco deve ser atravessado.

Uma vez que o grafo *G'* foi determinado, um ciclo de Euler pode ser construído através de uma adaptação do algoritmo de Fleury (referente ao caso orientado) ou pelo algoritmo de Aardenne- Ehrenfest e Bruijn (EISELT *et al*, 1995a), descrito abaixo:

Passo 1. Construa uma arborescência enraizada em um vértice qualquer v_r.

Passo 2. Rotule os arcos da seguinte maneira: ordene e rotule os arcos partindo de v_r de modo arbitrário; ordene e rotule os arcos fora de qualquer outro vértice consecutivo de modo arbitrário, até se chegar ao último arco da arborescência.

Passo 3. Determine um ciclo de Euler começando do arco com rótulo mais baixo partindo de um vértice arbitrário; sempre que se chegar a outro vértice, parte-se deste através de um arco ainda não percorrido e que tenha o rótulo mais baixo. O procedimento se encerra com um ciclo de Euler quando todos os arcos forem percorridos.

Como exemplo, seja o grafo orientado valorado da figura 8 no qual deverá ser determinado um ciclo com início e fim no vértice 1 e de extensão mínima.

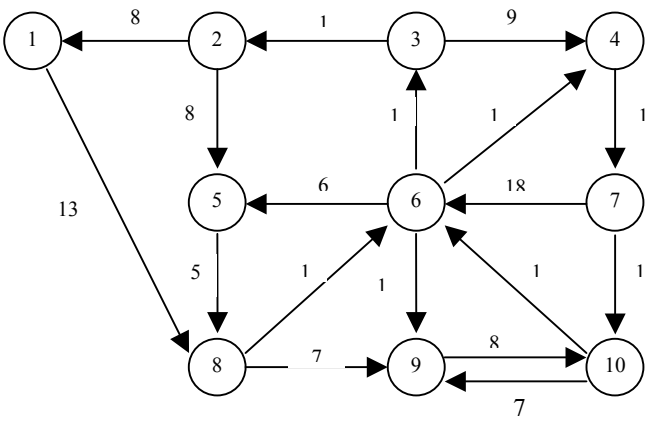


Figura 8 – Grafo orientado valorado

Neste grafo somente os vértices 1, 8 e 10 apresentam semigrau nulo. Portanto, será necessária a construção de um grafo aumentado *G'* no qual será determinado um ciclo de extensão mínima. Seja $I = \{4, 5, 9\}$ o conjunto dos vértices que representam os pontos de

fornecimento s_i e $J = \{2, 3, 6, 7\}$ o conjunto dos vértices que representam os pontos de demanda d_j. A figura 9 apresenta o grafo bipartido formado pelos elementos dos conjuntos I e J.

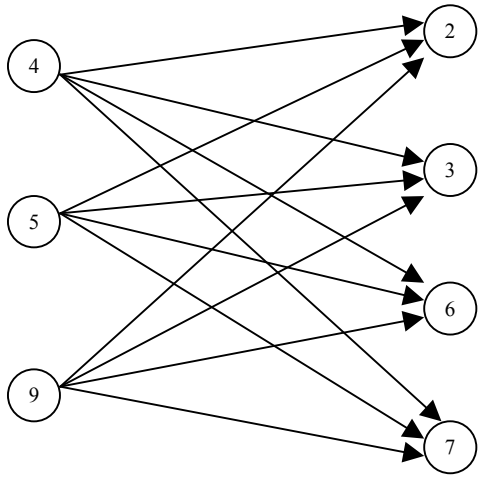


Figura 9 – Grafo bipartido

Os arcos que ligam os vértices de I e J tem como coeficientes de custo os valores dos caminhos mínimos em G. O problema de fluxo de custo mínimo pode ser formulado como:

Min z = $51x_{42} + 41x_{43} + 29x_{46} + 11x_{47} + 40x_{52} + 30x_{53} + 18x_{56} + 44x_{57} + 41x_{92} + 31x_{93} + 19x_{96} + 34x_{97}$

Sujeito a:

$x_{42} + x_{43} + x_{46} + x_{47} = 1$

$x_{52} + x_{53} + x_{56} + x_{57} = 1$

$x_{92} + x_{93} + x_{96} + x_{97} = 2$

$x_{42} + x_{52} + x_{92} = 1$

$x_{43} + x_{53} + x_{93} = 1$

$x_{46} + x_{56} + x_{96} = 1$

$x_{47} + x_{57} + x_{97} = 1$

$x_{ij} \geq 0, I \in \{4, 5, 9\}, J \in \{2, 3, 6, 7\}$

A solução ótima para o problema acima é z = 101 u. c. com $x_{47} = 1$, $x_{52} = 1$, $x_{93} = 1$ e $x_{96} = 1$. O grafo G' resultante é mostrado na figura 10 onde os arcos em linhas tracejadas representam

os segmentos de percurso ocioso. O ciclo de Euler em *G'* é dado pela seqüência 1, 8, 6, 5, 8, 9, 10, 9, 10, 6, 9, 10, 6, 4, 7, 10, 6, 3, 4, 7, 6, 3, 2, 5, 8, 6, 3, 2, 1 com um custo total de 125 u.c.

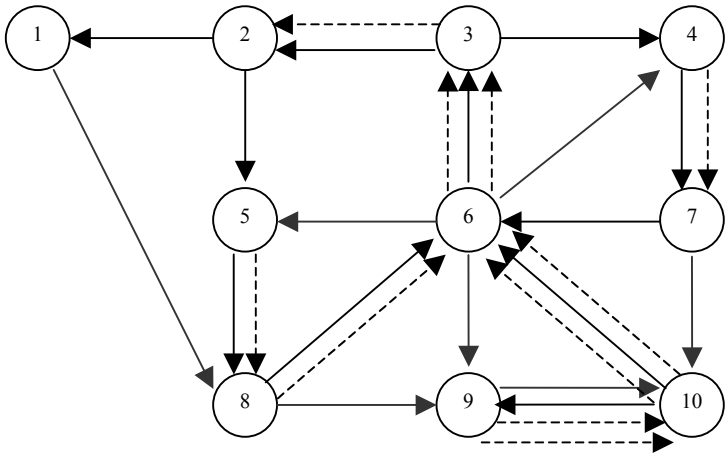


Figura 10 – Grafo orientado aumentado

Nos grafos mistos, um ciclo de Euler corresponde a um percurso em que os arcos e as arestas são atravessados exatamente uma vez. Este tipo de percurso é determinado em um grafo misto, se este for fortemente conexo e tiver todos os seus vértices com grau par e simétricos. A condição de simetria estabelece que em um vértice o número de arcos chegando deve ser igual ao número de arcos partindo. Se todas estas condições forem satisfeitas, o grafo é dito *equilibrado*, sendo possível obter a solução ótima para o PCC.

Se um grafo misto não for equilibrado, mas mantém a condição de fortemente conexo, a determinação de um ciclo dependerá da construção de um grafo aumentado *G'*, no qual alguns arcos e algumas arestas são replicados a custo mínimo. O percurso assim definido corresponderá a um ciclo de carteiro. Papadimitriou (1976) demonstrou que o PCC em grafos mistos não equilibrados é um problema *NP hard* (*Não Polinomial difícil*). Conceitualmente um problema de otimização combinatória é classificado como *NP hard* quando seu grau de complexidade aumenta exponencialmente à medida que aumenta o número de variáveis de decisão. Desse modo, a não ser em casos de pequenas instâncias, é inviável a obtenção de uma solução ótima em um tempo computacional aceitável.

O primeiro algoritmo para a resolução do PCC misto, desenvolvido a partir de uma abordagem heurística, foi apresentado por Edmonds e Johnson (1973). Tal algoritmo é composto de duas etapas. Na primeira, o grafo original é transformado em um grafo euleriano pela aplicação de um algoritmo de emparelhamento perfeito de peso mínimo (de maneira idêntica ao caso não orientado), sendo para isso ignorada a direção dos arcos (estes passam a ser arestas). Na segunda etapa o grafo resultante é tornado simétrico pela aplicação de um algoritmo de fluxo de custo mínimo. Uma vez que o grafo simétrico resultante não é necessariamente mantido como euleriano, podem ocorrer situações em que não é possível determinar um ciclo de carteiro, o que conduz a um problema sem solução.

Para contornar esta situação de indeterminação, Frederickson (1979) propôs uma modificação no algoritmo de Edmonds e Johnson acrescentando uma terceira etapa, na qual o grafo obtido após a execução da segunda etapa é tornado novamente euleriano. Este algoritmo, denominado *Mixed 1*, possui uma complexidade computacional da ordem de $O(\text{Max}\{|V|^3, |A| (\text{Max}\{|A|, |E|\})^2\})$, sendo V o número de vértices, A o número de arcos e E o número de arestas. O autor demonstrou também que o desempenho deste algoritmo tem no pior caso um limite de 2, ou seja, (solução Mixed 1)/(solução ótima) ≤ 2. A descrição do algoritmo é feita abaixo.

Passo 1. No grafo $G = (V, A \cup E)$ fortemente conexo identifique os vértices de grau ímpar. Determine os caminhos mínimos entre os vértices de grau ímpar, ignorando a direção dos arcos, e a seguir o emparelhamento perfeito de peso mínimo entre estes vértices. O grafo original é transformado então em um grafo aumentado pela inclusão dos arcos e arestas determinados pela solução do problema de emparelhamento perfeito.

Passo 2. Usando um algoritmo de fluxo de custo mínimo, torne o grafo aumentado simétrico. Seja $G' = (V, A' \cup E')$ o grafo resultante.

Passo 3. Identifique o conjunto dos vértices de grau ímpar em G'. Determine ciclos formados por caminhos alternantes (independente da direção dos arcos) em $A' \setminus A$ e E' sendo cada caminho amarrado em seus extremos por um vértice de grau ímpar. Como um ciclo é formado, seus arcos

podem ser tanto replicados como deletados e suas arestas são orientadas, de modo que o grafo resultante permaneça simétrico e se torne euleriano.

Ainda no mesmo trabalho, foram apresentados dois novos algoritmos para o PCC misto. O *Mixed 2* consiste basicamente em uma abordagem inversa do *Mixed 1*, sendo formado por duas etapas principais: na primeira o grafo é tornado simétrico e na segunda euleriano. A complexidade computacional deste algoritmo é a mesma do *Mixed 1*, bem como o seu desempenho no pior caso. O outro algoritmo consiste de uma abordagem mista na qual são utilizados os algoritmos *Mixed 1* e *Mixed 2* para a geração de duas soluções para o problema, sendo a seguir selecionada a melhor das duas. O desempenho deste procedimento tem no pior caso um limite de 5/3. Não há referencia no trabalho sobre a complexidade computacional do algoritmo. A descrição detalhada destes dois algoritmos pode ser encontrada no artigo original.

Pearn e Liu (1995) propuseram modificações nos algoritmos *Mixed 1* e *Mixed 2* a fim de serem obtidos melhores resultados finais. No algoritmo *Mixed 1* foi acrescentada uma quarta etapa, na qual são removidos os ciclos artificiais gerados com a aplicação dos algoritmos de emparelhamento perfeito de peso mínimo e fluxo de custo mínimo, sendo que a complexidade computacional permanece a mesma do algoritmo original. Já o *Mixed 2* apresenta um desempenho ruim quando os comprimentos das arestas são relativamente grandes e é aplicado o algoritmo de emparelhamento perfeito. Com o objetivo de melhorar a solução é adotada uma abordagem alternativa na qual, primeiramente alguns arcos são duplicados e, posteriormente arbitradas direções às arestas a fim de tornar o grafo euleriano. Foi considerada também pelos autores a estratégia mista, na qual é escolhida a melhor solução dentre aquelas geradas pelos algoritmos modificados. Pearn e Chou (1999) introduziram ainda outros procedimentos a estas versões modificadas do *Mixed 1*, *Mixed 2* e da abordagem mista a fim de serem obtidas soluções melhoradas para o PCC misto.

Mais recentemente Corberan *et al* (2002) desenvolveram um procedimento heurístico para o PCC misto baseado na metodologia *GRASP* (*Greedy Randomized Adaptative Search Procedure*). Testes computacionais com o emprego desta técnica feitos em um conjunto de 225 grafos gerados aleatoriamente, formados por mais de 200 vértices e 600 segmentos (arestas +

arcos) apresentaram melhores resultados do que aqueles obtidos a partir de outros algoritmos desenvolvidos para o PCC misto.

2.4.2 O Problema do Carteiro Rural

A exemplo do PCC visto no item anterior, o Problema do Carteiro Rural - PCR pode ser definido em grafos não orientados, orientados e mistos.

Um caso de resolução simples (em tempo polinomial) do PCR em redes orientadas e não orientadas ocorre quando os segmentos com demanda por serviço formam uma componente conexa. Porém quando esta condição não se verifica, é um problema *NP-hard* (LENSTRA e RINNOOY KAN, 1976 *apud* EISELT *et al*, 1995b)[1], sendo que a sua complexidade será proporcional ao número de componentes desconexas existentes em *G*. O PCR pode ser visto também como uma generalização do *Problema do Caixeiro Viajante* – PCV, no sentido de que cada instância do PCV pode ser convertida em uma instância do PCR.

Seja o grafo não orientado mostrado na figura 11, dado por $G(V, E_R \cup E)$, onde as arestas representadas em linha cheia correspondem aos segmentos de via com demanda por serviço. Os coeficientes de custo unitários associados às arestas de *G* representam a sua extensão em unidades de comprimento (u.c.). Deve ser determinado em *G* um percurso de extensão mínima, com início e fim no vértice 1 e que atravesse pelo menos uma vez cada elemento de E_R.

Neste exemplo, pode-se notar que os elementos de E_R formam uma componente conexa, ou seja, encontram-se ligados uns aos outros. Assim, a determinação de um percurso com início e fim em um mesmo vértice e que atravesse pelo menos uma vez cada um dos segmentos com demanda por serviço é feita de modo análogo ao caso do PCC em grafos não orientados descrita anteriormente. Porém, a determinação dos trechos de percurso ocioso no PCR pode contemplar a utilização de arestas contidas em *E* que representem o caminho mínimo entre vértices de grau ímpar. A solução ótima para este exemplo (figura 12) é dada pela seqüência 1, 6, 11, 12, 7, 12, 17, 18, 19, 18, 13, 8, 9, 14, 15, 10, 5, 10, 9, 4, 9, 8, 7, 2, 1 com uma extensão total de 24 u.c.

[1] LENSTRA, J. K.; RINNOOY KAN, A. H. G. On General Routing Problems. **Networks**, v. 6, p. 273-280, 1976.

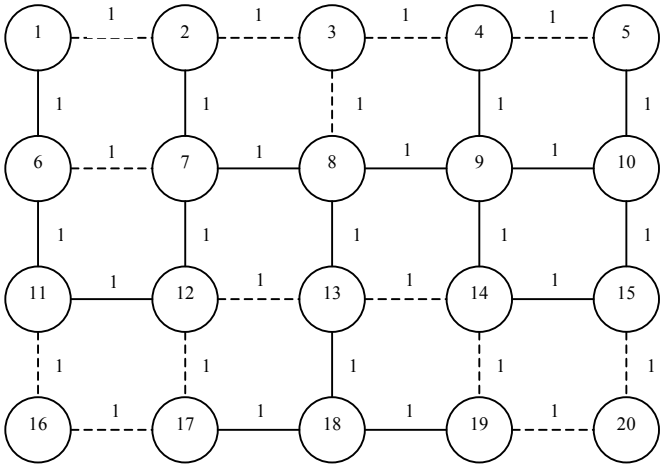


Figura 11. Grafo não orientado valorado com arestas requeridas formando uma componente conexa

Para os casos em que o conjunto E_R é formado por componentes desconexas, a solução do PCR em um grafo não orientado pode ser obtida com o emprego de heurísticas. Nesta linha de desenvolvimento Christofides *et al* (1981) *apud* Pearn e Wu (1995)[2] apresentaram um procedimento composto de 3 fases que envolvem a transformação do grafo onde cada componente é representada como um vértice; a conexão destes vértices através da resolução de um problema de árvore de peso mínimo; e por fim, tomando-se novamente o grafo original agora

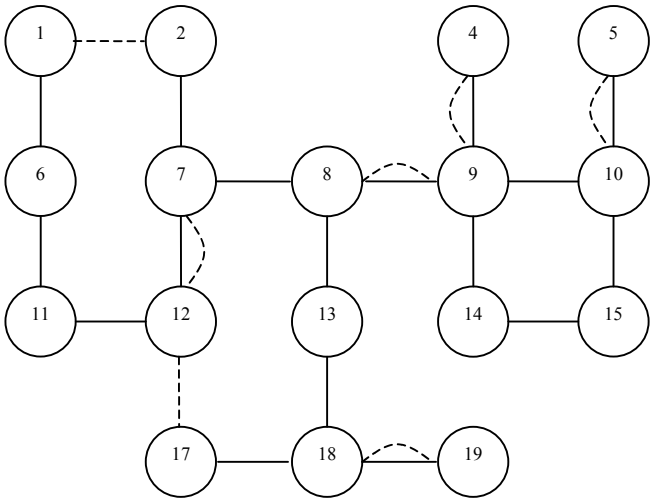


Figura 12. Resolução do PCR em um grafo não orientado

[2] CHRISTOFIDES, N.; CAMPOS, V.; CORBERÁN, A.; MOTA, E. An Algorithm for the Rural Postman Problem. **Imperial College Report**, London, 1981

conexo, a resolução de um problema de emparelhamento perfeito de peso mínimo. A complexidade deste algoritmo, limitada pelo problema de emparelhamento, é da ordem de $O(|V|^3)$. Se a desigualdade triangular de c_{ij} for satisfeita, a solução gerada por este algoritmo estará limitada em seu pior caso a 3/2 da solução ótima.

Pearn e Wu (1995) propuseram duas abordagens alternativas ao algoritmo de Christofides. Na primeira (chamada de versão *modificada*) é suprimida a fase de transformação do grafo, além de ser considerada uma nova distância $d(C_x, C_y) = \min_{ij} \{d(v_i, v_j) \mid v_i \in C_x, v_j \in C_y\} + \lambda$, sendo λ uma penalidade adicionada na definição das distâncias no grafo condensado G_C, e $d(x, y)$ o comprimento do caminho mínimo entre v_i e v_j no grafo original. Como diferentes valores de λ geram diferentes valores para a solução do PCR, pode ser escolhido um conjunto de valores de λ para serem geradas algumas soluções para o PCR e destas ser selecionada a melhor. A apresentação da seqüência dos passos que compõem esta abordagem é feita abaixo:

Fase I (Árvore de peso mínimo)

Passo 1. Seja C o conjunto que representa como vértices as componentes desconexas de E_R existentes em G, e G_C o respectivo grafo condensado. Uma aresta e_{xy} de G_C existe se houver uma aresta $e_{ij} \in G$ onde $v_i \in C_x$ e $v_j \in C_y$. Defina a distância entre cada par de vértices v_x e $v_y \in G_C$ como $d(v_x, v_y) = d(C_x, C_y) = \min_{i,j}\{\text{caminho mínimo}(v_i, v_j) \setminus v_i \in C_x, v_j \in C_y\} + \lambda$, com $\lambda = 0$ inicialmente. Aplique o algoritmo de árvore de peso mínimo sobre G_C. Seja $E_{To}(\lambda)$ o conjunto de arestas determinadas a partir da resolução do problema de árvore de peso mínimo.

Passo 2. Simplifique $E_{To}(\lambda)$ eliminando todas as cópias duplicadas de arestas que estejam em paralelo. Denomine o conjunto de arestas resultante como $E_T(\lambda)$.

Fase II (Emparelhamento de peso mínimo)

Resolva o PCC sobre $G \cup E_T(\lambda)$ aplicando o algoritmo de emparelhamento perfeito de peso mínimo, a fim de ser obtido um grafo euleriano. Um ciclo de carteiro pode então ser definido a

partir deste grafo euleriano. Seja $E_M(\lambda)$ o conjunto de arestas obtidas a partir da resolução do problema de emparelhamento. A solução para o PCR é então dada por $G(\lambda) = G \cup E_T(\lambda) \cup E_M(\lambda)$.

Fase III (Iterações com valores de parâmetro variáveis)

Repita as fases I e II para um conjunto de valores pré-determinados $\{\lambda_1, \lambda_2, ..., \lambda_n\}$ para o parâmetro de penalidade λ, gerando um conjunto de soluções para o PCR em G. Selecione a melhor (de menor custo) entre todas aquelas obtidas a partir desta abordagem.

Na segunda abordagem é adotada uma ordem inversa no algoritmo, suprimindo novamente a fase de transformação do grafo e adicionando-se a penalidade λ. Testes comparativos compreendendo 200 problemas demonstraram a capacidade das abordagens alternativas de alcançarem melhores resultados em relação ao algoritmo original, no tocante a média percentual acima do limite inferior; a classe média entre os três algoritmos; o número de problemas atingindo a melhor solução; e o número de problemas atingindo o limite inferior.

Cook *et al* (1998) apresentaram quatro diferentes procedimentos para melhoria da solução inicial do PCR obtida a partir da utilização das fases II e III do algoritmo de Christofides. Nos procedimentos denominados *Force One Edge* e *Force Two Edges* são buscadas, respectivamente, uma e duas arestas em E que possam ligar duas componentes desconexas de E_R gerando o percurso de menor custo. Já no procedimento *Iterated Force One Edge* esta busca é feita de modo iterativo, e envolve a escolha de uma aresta entre duas componentes. O último procedimento apresentado utiliza a técnica dos *Algoritmos Genéticos*. Testes comparativos entre os quatro procedimentos apontaram o *Iterated Force One* como aquele capaz de gerar os melhores resultados para o PCR em relação à extensão dos percursos.

A aplicação do método de simulação *Monte Carlo* na resolução do PCR em redes não orientadas é apresentada no trabalho de Córdoba *et al* (1998). Em testes realizados através da execução de 26 problemas, 24 atingiram a solução ótima, sendo o resultado global melhor do que aquele alcançado com o emprego do algoritmo de Christofides.

Letchford e Eglese (1998) apresentaram um algoritmo de solução ótima para o PCR com janelas de tempo. Neste caso, a estratégia de solução foi baseada no uso de inequações válidas como planos de corte, técnica esta aplicada anteriormente na resolução do PCV.

Assim como no caso não orientado, o PCR definido em um grafo orientado $G(V, A_R \cup A)$ terá resolução semelhante àquela vista anteriormente para o PCC orientado se os elementos de A_R formarem uma componente conexa. Se for verificada em G a existência de componentes desconexas, o PCR deverá ser resolvido em um grafo modificado.

Christofides *et al* (1986) *apud* Eiselt *et al* (1995b)[3] propuseram a seguinte heurística para resolução do PCR em grafos orientados:

Passo 1. Construa a árvore de peso mínimo enraizada em um vértice arbitrário e que conecte todas as componentes C_r em G. Adicione ao grafo original as arestas obtidas na resolução da árvore de peso mínimo, resultando em um grafo G_R.

Passo 2. Assim como no caso orientado do PCC, construa um grafo euleriano a partir de G_R através da adição de arcos de mínimo custo de modo que o número de arcos chegando em cada vértice seja igual ao número de arcos partindo.

Passo 3. Determine um ciclo de Euler no grafo aumentado obtido no passo anterior.

O PCR definido em uma rede mista $G(V, A_R \subset A \cup E_R \subset E)$ consiste na generalização de dois problemas *NP hard* que são o PCC misto e o PCR (não orientado e orientado). Conseqüentemente, o PCR misto é também *NP hard* (Córberan *et al*, 2000).

Laporte (1997) apresentou uma abordagem de solução do PCR misto através da sua transformação no PCV. Uma vez feita à transformação, duas estratégias de resolução para o PCV foram empregadas comparativamente: um método de solução ótima (*branch-and-bound*) e um

[3] CHRISTOFIDES, N.; CAMPOS, V.; CORBERÁN, A.; MOTA, E. An Algorithm for the Rural Postman Problem on a Directed Graph. **Mathematical Programming Study**, v. 26, p. 155-166, 1986.

método heurístico (algoritmo de Karp). Os resultados computacionais obtidos demonstraram que em redes esparsas, com um número reduzido de arestas ($E < 30$), a solução do problema pode atingir o valor ótimo.

No trabalho de Córberan *et al* (2000) foram apresentados dois procedimentos heurísticos para resolução do PCR misto: um algoritmo construtivo baseado na resolução de problemas de fluxo de custo mínimo e emparelhamento perfeito de peso mínimo; e um procedimento de busca local utilizando diferentes elementos da *Busca Tabu*. Experimentos computacionais realizados com um conjunto de 270 instâncias com $20 \leq | V | \leq 100$, $15 \leq | E | \leq 220$, $5 \leq | E_R | \leq 150$, $55 \leq |A| \leq 350$, $5 \leq | A_R | \leq 200$ geradas aleatoriamente demonstraram um bom desempenho dos dois procedimentos em alcançar valores bastante próximos ao ótimo em pequenos tempos de processamento.

Córberan *et al* (2002) apresentaram uma generalização do PCR misto consistindo na associação de uma penalidade não negativa a cada movimento de curva feito no roteiro (dobrar a direita ou à esquerda e retornos), bem como em relação a movimentos de curva proibidos. Neste trabalho foi demonstrada a possibilidade da resolução do problema através de sua transformação em um PCV assimétrico, de acordo com a estratégia de solução do PCR misto apresentada por Laporte (1997). Um procedimento heurístico baseado no método apresentado por Córberan *et al* (2000) também foi apresentado. Testes computacionais para avaliação da eficiência do procedimento heurístico desenvolvido e do procedimento de transformação do problema no PCV assimétrico foram realizados em um conjunto de 279 instâncias de vários tamanhos e tipos. A comparação feita a partir dos resultados obtidos favoreceu o procedimento heurístico, devido aos pequenos desvios em relação à solução ótima em instâncias de maior tamanho (principalmente quando $E_R > 30$).

2.5 VARIANTES DO PROBLEMA DE ROTEIRIZAÇÃO EM ARCOS

Além do PCC e do PCR definidos em grafos não orientados, orientados e mistos, outras variantes do Problema de Roteirização em Arcos são referidas pela literatura.

O *Hierarchical Chinese Postman Problem* - HCPP, introduzido por Dror *et al* (1987), é uma variante do PCC no qual são formados conjuntos de arestas (também chamados de *partições*) e uma relação de precedência é definida sobre eles. Em termos práticos o HCPP descreve situações onde o atendimento de alguns segmentos de uma rede viária deve ser feito antes dos demais.

Se existir uma relação de precedência linear entre a ordem de atendimento dos conjuntos de arestas, estes podem ser ordenados em c classes onde $E_1, E_2, ..., E_c$ ($E_1 \cup E_2 \cup ... E_c = E$, $E_i \cap E_j = \phi \ \forall \ i, j \in \{1, ..., c\}, i \neq j$). Além disso, nenhuma aresta em E_i pode ser atravessada enquanto existir uma aresta em E_{i-1} não atravessada.

Dror *et al* (1987) demonstraram que o HCPP é um problema *NP hard*. Entretanto, este problema pode ser resolvido em tempo polinomial ($O(c|V|^5)$) se existir uma relação de precedência linear entre os conjuntos de arestas e estes forem conexos. Mais recentemente, Ghiani e Improta (2000) propuseram um algoritmo de solução ótima para o HCPP baseado na resolução de um problema de emparelhamento. Já Cabral *et al* (2003) apresentam uma abordagem de resolução para o HCPP a partir de sua transformação no PCR.

O *Windy Postman Problem* - WPP consiste em uma generalização do PCC, no qual cada segmento possui dois coeficientes de custo de valores distintos, dados de acordo com o sentido em que o segmento é atravessado. Deste modo, a denominação *Windy Postman Problem* refere-se a uma pessoa que realiza um percurso a favor do vento, o que implica em um custo mais baixo. Uma outra forma de interpretação para esta denominação está relacionada ao custo de se atravessar segmentos em aclives (maior custo) e declives (menor custo).

Em relação a sua complexidade, o WPP é *NP hard* (GUAN, 1984), sendo computacionalmente equivalente ao PCC misto. Deste modo, o WPP pode ser transformado no PCC misto e resolvido como tal.

Procedimentos heurísticos para resolução do WPP foram apresentados por Guan (1984) e Win (1989) *apud* Pearn e Li (1994)[4], ambos com complexidade de $O(|V|^3)$. Pearn e Li (1994) apresentaram uma versão modificada para o algoritmo de Guan e uma abordagem inversa ao algoritmo de Win. A comparação do desempenho dos quatro algoritmos foi feita através de testes computacionais em 50 problemas com $8 \leq |V| \leq 50$ e $14 \leq |E| \leq 377$. Os resultados obtidos demonstraram o algoritmo de Win como o de melhor desempenho em relação à média percentual acima do limite inferior; a classe média entre os três algoritmos; ao número de problemas atingindo a melhor solução; e ao número de problemas atingindo o limite inferior. Ainda neste trabalho, testes com um conjunto extra de 240 problemas com significativa percentagem de vértices de grau ímpar ($33\% \leq V_0 \leq 100\%$) demonstraram melhores resultados da abordagem inversa do algoritmo de Win em relação aos demais no tocante a quantidade de problemas atingindo a melhor solução (73%).

O *Stacker Crane Problem* - SCP pode ser visto como um caso do PCR misto, no qual os arcos representam os trechos com demanda por serviço. O SCP consiste então em determinar um percurso de custo mínimo incluindo cada elemento de A pelo menos uma vez. A denominação dada ao SCP pode ser entendida a partir da consideração dos arcos como movimentos feitos por um guindaste, exatamente uma vez em uma certa direção. Se o custo c_{ij} em cada arco é zero, o SCP é equivalente ao PCV, sendo assim um problema *NP hard* (EISELT *et al*, 1995b).

Dois procedimentos heurísticos para resolução do SCP foram propostos por Frederickson *et al* (1978) *apud* EISELT *et al* (1995b)[5]. Nestes dois algoritmos, denominados *Largearcs* e *Smallarcs*, a rede representada por G deve satisfazer as seguintes propriedades: cada vértice é incidido por pelo menos um arco, e os coeficientes c_{ij} das arestas devem satisfazer a desigualdade

[4] WIN, Z. On the Windy Postman Problem on Eulerian Graphs. **Mathematical Programming**, v. 44, p. 97-112, 1989.
[5] FREDERICKSON, G.; HECHT, M.; KIM, C. Approximation Algorithms for Some Postman Problems. **SIAM Journal of Computing**, v. 7, p. 178-193, 1978.

triangular. Se G não satisfaz estas duas propriedades pode ser transformado em um grafo equivalente G'. O problema é resolvido então em G' e a solução pode ser interpretada em termos do grafo original G. A complexidade dos dois algoritmos é definida em $O(\max\{|V|^3, |A|^3\})$.

De maneira semelhante ao caso do PCR misto, Laporte (1997) propôs a resolução do SCP através de sua transformação em uma versão generalizada do PCV, possibilitando assim o emprego de métodos de solução ótima, bem como de métodos heurísticos eficientes. Permanece porém, a mesma limitação observada no caso do PCR misto referente ao número de arestas existentes em G ($E \leq 30$).

2.6 O PROBLEMA DE ROTEIRIZAÇÃO EM ARCOS CAPACITADO

O Problema de Roteirização em Arcos Capacitado - PRAC, introduzido por Golden e Wong (1981), consiste em uma generalização do Problema de Roteirização em Arcos cujo modelo pode tomar em consideração restrições de cunho operacional. Estas restrições podem ser o limite de carregamento das unidades de trabalho (veículos, equipamentos ou pessoas) e o tempo máximo de duração das jornadas de trabalho, além de outras situações mais específicas como janelas de tempo e frotas ou equipes heterogêneas. É descrita assim, de maneira mais efetiva, a realidade operacional dos serviços modelados como Problemas de Roteirização em Arcos em todas as suas subclasses (PCC e PCR) e variantes (HCPP, WPP e SCP).

A definição do PRAC é feita a partir de uma rede de atendimento representada por um grafo conexo dado por $G(V, E_R \subseteq E \cup A_R \subseteq A)$. A cada segmento da rede são associados: um coeficiente $c_{ij} > 0$ referente ao custo de seu percurso; e um coeficiente $q_{ij} \geq 0$ quantificando a demanda (em unidades relacionadas a tempo, peso ou volume) por serviço aí existente. Seja ainda $v_D \in V$ o vértice que representa o *depósito*, ou seja a base operacional a partir da qual são despachadas as unidades de trabalho alocadas ao atendimento das demandas em G. Cada unidade de trabalho possui uma capacidade máxima dada por W.

O PRAC consiste na determinação de um conjunto de roteiros de custo mínimo, nos quais os segmentos com demanda por serviço são percorridos pelo menos uma vez e cuja demanda total não exceda a capacidade W de uma unidade de trabalho. Os roteiros devem ser definidos

então de modo que: (1) cada um dos segmentos de *G* com demanda seja atendido por exatamente uma unidade de trabalho; (2) a demanda existente em um segmento não exceda a capacidade *W* de uma unidade de trabalho; e (3) o custo total de atendimento de *G* seja mínimo.

Quando $W \geq \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij}$, o PRAC se reduz ao PCC se $E_R \cup A_R = E \cup A$, e ao PCR se $E_R \cup A_R \subset E \cup A$.

Golden e Wong (1981) apresentaram a seguinte formulação de programação inteira para o PRAC:

$$\text{Minimizar } z = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{p=1}^{k} c_{ij} x_{ij}^{p} \qquad (13)$$

sujeito a:

$$\sum_{k=1}^{n} x_{ki}^{p} - \sum_{k=1}^{n} x_{ik}^{p} = 0 \qquad \text{para } i = 1, ..., n \text{ e } p = 1, ..., k \qquad (14)$$

$$\sum_{p=1}^{k} (l_{ij}^{p} + l_{ji}^{p}) = \left\lceil \frac{q_{ij}}{W} \right\rceil \qquad \text{para } e_{ij} \in E_R \text{ ou } a_{ij} \in A_R \qquad (15)$$

$$x_{ij}^{p} \geq l_{ij}^{p} \qquad \text{para } e_{ij} \in E \text{ ou } a_{ij} \in A \text{ e } p = 1, ..., k \qquad (16)$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} l_{ij}^{p} q_{ij} \leq W \qquad \text{para } p = 1, ..., k \qquad (17)$$

$$\left.\begin{array}{ll} \sum_{i \in Q'} \sum_{j \in Q'} x_{ij}^{p} - n^2 y_{1q'}^{p} \leq |Q'| - 1 & \text{para } p = 1, ..., k; \\ \sum_{i \in Q'} \sum_{j \notin Q'} x_{ij}^{p} + y_{2q'}^{p} \geq 1 & q' = 1, \ldots, 2^{n-1}\text{-}1; \text{ e cada} \\ y_{1q'}^{p} + y_{2q'}^{p} \leq 1; y_{1q'}^{p}, y_{2q'}^{p} \in \{0,1\} & \text{subconjunto não vazio } Q' \text{ de } \{2, 3, ..., n\} \\ x_{ij}^{p}, l_{ij}^{p} \in \{0,1\} & \end{array}\right\} \qquad (18), (19)$$

onde:

k é o número de unidades de trabalho;

x_{ij}^{p} é uma variável binária que assume o valor 1 se o segmento compreendido entre v_i e v_j é atravessado pela unidade p e 0 caso contrário;

l_{ij}^{p} é uma variável binária que assume o valor 1 se a unidade p atende o segmento compreendido entre v_i e v_j e 0 caso contrário.

No modelo matemático acima apresentado, a restrição (14) garante a continuidade do percurso; a restrição (15) estabelece que cada segmento com demanda positiva seja atendido exatamente uma vez; a restrição (16) garante que cada um dos segmentos da rede possa ser atendido pela unidade de trabalho p somente se esta atravessar tal segmento; a restrição (17) assegura a não violação da capacidade W de cada unidade de trabalho; a restrição (18) proíbe a construção de sub-roteiros ilegais (ou seja, pequenos roteiros desconexos do roteiro principal); e a restrição (19) a integralidade da solução.

Como o PRAC envolve restrições operacionais que limitam a capacidade de atendimento das unidades de trabalho, sua resolução de maneira exata é um tanto difícil. Golden e Wong (1981) demonstraram que mesmo uma versão aproximada de 0,5 do PRAC, na qual se busca a determinação de uma solução com um custo menor do que 1,5 vezes o valor da solução ótima, é um problema *NP hard*.

Devido ao seu elevado grau de complexidade, o PRAC encontra nos métodos aproximativos uma alternativa eficiente de resolução. No contexto do PRAC, de modo particular, estes métodos podem ser subdivididos em 3 categorias básicas (EISELT *et al*, 1995b): (1) os construtivos simples; (2) os construtivos de duas fases; e (3) os de melhoria.

Na categoria dos métodos construtivos simples, de um modo geral, os roteiros são formados de maneira sequencial através da agregação de segmentos. Posteriormente estes roteiros podem ser agrupados segundo um critério de ganho, de modo semelhante ao método das economias de Clarke e Wright (ver BODIN *et al*, 1983). No quadro 3 são citados os algoritmos pertencentes a esta categoria, a respectiva ordem de complexidade e a referência na literatura.

Testes computacionais comparativos entre estes algoritmos são apresentados por Pearn (1989 e 1991). Em grafos densos, com pequenas demandas nas arestas, o *Modified construct-strike* obteve as melhores soluções (proximidade em relação ao limite inferior) em um conjunto de problemas-teste com $11 \leq |V| \leq 17$ e densidade de arestas variando entre 70% e 100%. Já em grafos pouco densos, com grandes demandas nas arestas, o *Augment-insert* atingiu os melhores resultados, também considerando a proximidade em relação ao limite inferior, em aplicações feitas em um conjunto de problemas-teste com $13 \leq |V| \leq 27$ e densidade de arestas variando entre 15% e 30%.

Quadro 3 – Algoritmos pertencentes à categoria dos métodos construtivos simples

Algoritmo	Complexidade	Referência
Construct-strike	$O(\|E\|\|V\|^3)$	Christofides, 1973 *apud* Eiselt *et al*, 1995b[6]
Modified construct-strike	$O(\|E\|\|V\|^4)$	Pearn, 1989
Path-scanning	$O(\|V\|^3)$	Golden *et al*, 1983
Random path-scanning	$O(\|V\|^3)$	Pearn, 1989
Augment-merge	$O(\|V\|^3)$	Golden *et al*, 1983
Augment-insert	$O(\|V\|^3)$	Pearn, 1991

Os métodos construtivos de duas fases compreendem duas estratégias básicas de solução: *agrupar primeiro – roteirizar depois* e *roteirizar primeiro – agrupar depois*. Em ambos os casos a fase de agrupamento consiste no particionamento de uma rede de atendimento, procedimento este que apresenta certas particularidades. Assim, os métodos construtivos de duas fases serão vistos com maiores detalhes no capítulo 3 que trata do Particionamento de Redes de Transporte.

A terceira categoria engloba os métodos de melhorias, os quais são aplicados sobre soluções geradas a partir de um dos dois métodos referidos anteriormente, tendo como objetivo produzir soluções finais com um valor o mais próximo possível do limite inferior que caracteriza um determinado problema.

[6] CHRISTOFIDES, N. The Optimal Traversal of a Gragh. **Omega**, v. 1, p. 719-732, 1973.

Para execução deste processo de melhoria, podem ser empregados métodos baseados na troca de segmentos entre ciclos adjacentes. Isto é feito através da adaptação de técnicas como o 2-opt, o 3-opt e o k-opt, utilizados nos Problemas de Roteirização em Nós. Os aspectos conceituais e de implementação destas técnicas são apresentados nos trabalhos de Lin e Kernighan (1973), Bodin *et al* (1983) e Helsgaun (2000). Pode ser citada, ainda a técnica *interativa*, onde o processo de melhoria é orientado pela intervenção do usuário do sistema através de seu julgamento pessoal.

O processo de melhoria pode envolver também a utilização de métodos generalistas, conhecidos como *metaheurísticas*. Fazem parte destes métodos técnicas como a *Busca Tabu* (*Tabu Search*), a *Têmpera Simulada* (*Simulated Anneling*), os *Algoritmos Genéticos* (*Genetic Algorithms*), a *Busca Dispersa* (*Scatter Search*) e a *Colônia de Formigas* (*Ant Colony*), além das estratégias *híbridas* que funcionam com o uso combinado de duas ou mais destas técnicas. Nos trabalhos de Laguna (2003) e Marti (2003) são apresentadas excelentes revisões sobre conceitos fundamentais dos métodos generalistas e sua aplicação em problemas de otimização combinatória, como é o caso do PRAC.

No quadro 4 são citados os trabalhos que descrevem a aplicação dos métodos de melhorias no PRAC.

Quadro 4 – Trabalhos relatando a aplicação de métodos de melhorias na resolução do PRAC

Referência	**Técnica empregada**
Eglese (1994)	Têmpera Simulada
Li e Eglese (1996)	Interativa
Greistorfer (2002)	Híbrida (Busca Tabu e Busca Dispersa)
Beullens *et al* (2003)	Busca Local Guiada

Além das 3 categorias básicas de métodos de resolução do PRAC (construtivos simples, construtivos de duas fases, e melhorias), deve ser feita também referência a alguns trabalhos em que são apresentados métodos alternativos. Dror e Langevin (1997) apresentam uma abordagem

de resolução do PRAC em grafos orientados através de sua transformação em uma versão generalizada do PCV. Também considerando o caso de grafos orientados, Wang e Wen (2002) apresentam um método de resolução do PRAC com janelas de tempo, utilizado a lógica *fuzzy*. Já no trabalho de Belenguer e Benavent (2003) é apresentado o desenvolvimento de um algoritmo de *planos de corte* para o cômputo de limites mínimos para o PRAC.

2.7 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Os Problemas de Roteirização em Arcos, cuja origem remonta a proposição da Teoria dos Grafos feita por Leonhard Euler em 1736, tem sido um importante campo de pesquisa na área de otimização nos últimos 40 anos principalmente no que diz respeito ao desenvolvimento de modelos de resolução. Dois elementos contribuíram decisivamente para isso: o desenvolvimento de modelos de programação matemática, principalmente os de conexão e fluxos em rede; e o desenvolvimento de recursos computacionais no que tange a velocidade de processamento e a capacidade de memória, além das ferramentas de visualização e de interação do usuário com o modelo.

O processo de resolução dos Problemas de Roteirização em Arcos exige uma forma abstrata de representação das redes viárias, que pode ser feita através dos grafos (ou multigrafos) não orientados, orientados e mistos. Nestas estruturas podem ser definidos 4 tipos de percursos que são o *ciclo simples*, o *ciclo de Euler*, o *ciclo de carteiro* e o *caminho*.

A forma de ocorrência das demandas nos segmentos que compõem a rede de atendimento implica em uma subclassificação dos Problemas de Roteirização em Arcos que são o *Problema do Carteiro Chinês* (PCC) e o *Problema do Carteiro Rural* (PCR).

No PCC todos os segmentos da rede de atendimento apresentam demanda e devem ser percorridos pelo menos uma vez. Se o grafo correspondente for do tipo não orientado ou orientado, o problema pode ser resolvido em tempo polinomial com o emprego de algoritmos de solução ótima. Já no caso da rede mista, o problema apresenta um grau de complexidade exponencial, sendo classificado como *NP-hard* e exigindo para sua resolução o emprego de

métodos aproximativos onde possa ser obtida uma boa solução em um tempo de processamento aceitável.

O PCR descreve a situação em que alguns segmentos da rede de atendimento apresentam demanda por serviço e devem ser necessariamente percorridos pelo menos uma vez. O PCR pode ser definido em grafos não orientados, orientados e mistos. Casos mais simples, com resolução em tempo polinomial, aparecem nos grafos não orientados e orientados onde os segmentos com demanda formam uma componente conexa. O PCR é *NP-hard* nos casos de redes mistas e de segmentos com demanda formando componentes desconexas.

Além das duas subclasses básicas, os Problemas de Roteirização em Arcos apresentam variantes que são o *Hierarchical Chinese Postman Problem* (HCPP), o *Windy Postman Problem* (WPP) e o *Stacker Crane Problem* (SCP). Em relação às particularidades dos percursos que são descritas por estas variantes estabelece-se uma priorização no atendimento dos segmentos da rede no caso do HCPP; uma priorização do sentido de percurso dos segmentos no caso do WPP e a discriminação dos segmentos com demanda (os arcos) no caso do SCP.

Na grande maioria dos serviços modelados como Problemas de Roteirização em Arcos, é necessário considerar restrições de capacidade, como o tempo máximo de jornada de trabalho e o limite de carregamento das unidades que compõem uma frota ou equipe. Objetivando possibilitar o tratamento deste tipo de situação, foi introduzido o Problema de Roteirização em Arcos Capacitado (PRAC). Devido ao elevado grau de complexidade apresentado pelo PRAC, sua resolução é feita a partir do emprego de métodos aproximativos, que são compreendidos por três categorias básicas: os construtivos simples, os construtivos de duas fases e os de melhoria. No caso específico dos métodos construtivos de duas fases, o processo de resolução envolve o Problema de Particionamento de Redes de Transporte, tema este que será discutido no próximo capítulo.

3 PROBLEMAS DE PARTICIONAMENTO DE REDES DE TRANSPORTE

3.1 INTRODUÇÃO

De acordo com a descrição feita no capítulo anterior, no PRAC é representada uma situação em que uma única unidade de trabalho (veículo, equipamento ou pessoa) não é capaz de atender a todas as demandas existentes em uma rede de serviço devido à existência de restrições relacionadas à capacidade, como o limite de carregamento (peso ou volume, por exemplo) e o tempo máximo da jornada de trabalho. Deste modo, é necessário que uma frota ou equipe composta por duas ou mais unidades de trabalho seja alocada a rede de atendimento, sendo respeitadas, contudo, as restrições impostas.

Entre as 3 categorias dos métodos aproximativos aplicados na resolução deste problema encontra-se a dos construtivos de duas fases. Como a sua própria denominação indica, estes métodos possuem como principal característica à divisão do processo de solução em duas fases distintas: uma de determinação dos segmentos que serão designados a cada unidade de trabalho, ou fase de particionamento; e uma de determinação da seqüência em que estes segmentos devem ser percorridos, ou fase de roteirização.

A ordem em que estas duas fases são dispostas no processo de resolução do problema faz com que os métodos construtivos de duas fases compreendam duas estratégias básicas que são:

a) Roteirizar primeiro – agrupar depois (*route first – cluster second*), e;

b) Agrupar primeiro – roteirizar depois (*cluster first – route second*).

Embora ambas estratégias contemplem as fases de particiomento e roteirização, ainda que de maneira alternada, são observadas características distintas no processo construtivo e nos resultados obtidos. Estas características serão abordadas em maiores detalhes no item 3.3.

Uma observação a ser feita é que, independente da estratégia adotada, a fase de roteirização obedece aos mesmos princípios utilizados na resolução do PCC, do PCR e demais variantes dos Problemas de Roteirização em Arcos. O particionamento, porém, possui certas peculiaridades em termos conceituais que são apresentados a seguir.

3.2 ASPECTOS CONCEITUAIS DO PARTICIONAMENTO DE REDES DE TRANSPORTE

No contexto do PRAC, o particionamento de redes de transporte consiste na definição de dois ou mais conjuntos formados por segmentos de via, sendo que cada um destes conjuntos é designado a uma unidade de trabalho pertencente a uma frota ou equipe.

Seja um grafo não orientado *G(V, E)* que representa uma rede de atendimento, e cujo conjunto *E* de arestas deva ser particionado em um número *k* de elementos. O índice *k* indica a quantidade de conjuntos de arestas, também chamados de *partições* (ou *clusters*), que são dados por $SG_1, SG_2, ..., SG_k$, onde $SG_i \subseteq E$, $SG_i \cap SG_j = \varnothing$ para $i \neq j$ e $\cup^{k}_{i=1} SG_i = E$.

Na figura 13 é mostrado um grafo não orientado formado por quatro partições (identificadas pelo tipo de traçado das arestas), onde $SG_1, SG_2, SG_3, SG_4 \subset E$. Pode-se notar nesta figura que cada aresta pertence a uma e somente uma partição SG_i.

As condições referidas acima são extensíveis ao caso dos grafos orientados e mistos. Valem também para a situação descrita no PCR, onde $E_R \subseteq E \cup A_R \subseteq A$.

Os critérios que orientam o particionamento de redes no PRAC podem variar de um caso para outro, em decorrência das particularidades apresentadas por cada problema. De um modo geral, esta variabilidade nos critérios se deve às características da rede, a forma de distribuição das demandas e as rotinas operacionais inerentes a cada serviço.

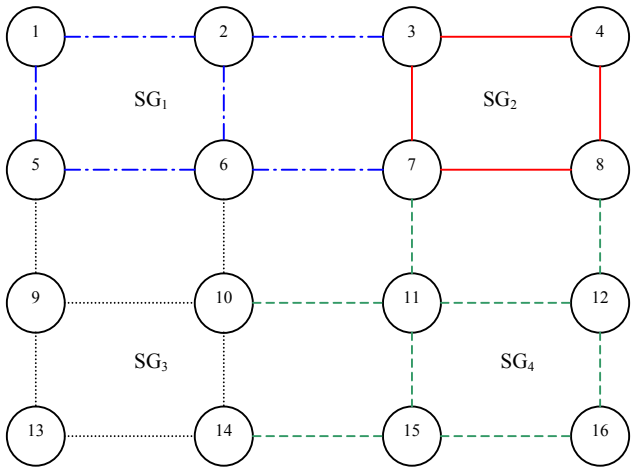


Figura 13 – Grafo não orientado formado por 4 partições

Porém, mesmo considerando possíveis variações existentes no critério de particionamento, valem certos princípios gerais que caracterizam cada uma das estratégias empregadas nos métodos construtivos de duas fases, apresentadas no próximo item.

3.3 ESTRATÉGIAS ADOTADAS NO PARTICIONAMENTO DE REDES DE TRANSPORTE

Na estratégia de roteirizar primeiro – agrupar depois a fase inicial compreende a construção de um ciclo, no qual todos os segmentos que apresentam demanda em *G* são percorridos pelo menos uma vez. Este ciclo, também chamado de *roteiro gigante*, é obtido, portanto, através do relaxamento da(s) restrição(s) associada(s) ao problema.

Na fase seguinte o roteiro gigante é particionado em uma série de roteiros viáveis que possuem extensão compatível com o limite dado pela(s) restrição(s) considerada(s). Devido às características do processo construtivo adotado na fase de particionamento, cada um dos roteiros viáveis resultantes consistirá em um caminho entre dois vértices distintos v_i e v_j.

Devido às particularidades do processo resolutivo adotado na estratégia de roteirizar primeiro – agrupar depois, algumas observações devem ser feitas a respeito do limite inferior e superior da solução a ser obtida. São considerados aqui os casos do limite de carregamento (peso

ou volume) e tempo máximo para jornada de trabalho, por serem estas as restrições mais freqüentes no PRAC.

Seja C_T a demanda total existente em uma rede de atendimento determinada a partir do somatório dos coeficientes q_{ij} e n o valor da restrição associada a cada uma das unidades de trabalho. O número de partições que devem ser construídas em G é definido por:

$$k = \frac{C_T}{n} \qquad (20)$$

onde k é um número inteiro.

O valor de k assim obtido representa o limite inferior para a solução do problema.

Nos casos do PCC definido em um grafo euleriano e do PCR conexo também definido em um grafo euleriano, o valor de C_T corresponderá ao tempo efetivo total de percurso resultando em um valor único para k quando houver restrição ao tempo máximo para jornada de trabalho.

Já em duas outras situações o valor de k não é único. A primeira ocorre no PCC definido em um grafo não euleriano, onde a solução compreende também alguns segmentos que representam os trechos de percurso ocioso. A segunda ocorre no PCR definido em um grafo não euleriano formado ou não por componentes desconexas, onde são considerados também os segmentos de percurso ocioso.

Nestas duas situações o valor de k estará contido em um intervalo dado por $[k_i, k_S]$ onde k_i representa o limite inferior e k_S o limite superior. Estes valores são obtidos a partir das seguintes expressões:

$$k_i = \frac{C_T}{n}; \qquad (21)$$

$$k_S = \frac{C_{TS}}{n} \qquad (22)$$

onde:
C_T é o tempo total de atendimento da rede considerando o somatório dos trechos de percurso efetivo;
C_{TS} é o tempo total de atendimento da rede considerando o somatório dos trechos de percurso efetivo e dos trechos de percurso ocioso.

Quando a restrição do problema estiver relacionada ao limite de carregamento, o valor de C_T será determinado considerando-se somente os segmentos onde $q_{ij} > 0$. Obtém-se assim um limite inferior semelhante aquele do PCC em um grafo euleriano e do PCR conexo em um grafo euleriano.

Ainda que a solução obtida esteja compreendida entre um limite mínimo e um limite máximo, em alguns casos, seu valor pode ser melhorado mediante o remanejamento de segmentos de uma partição para outra ou através da alteração do ponto de início dos roteiros contidos em *G*. Este processo de melhoria pode ser desenvolvido com o emprego das técnicas citadas no capítulo 2 (item 2.6).

Na figura 14 é mostrado um exemplo de aplicação da estratégia de roteirizar primeiro – agrupar depois em um grafo valorado não orientado. Os coeficientes de custo indicados nas arestas representam o tempo de atendimento (em horas) de cada segmento de via. Deseja-se determinar o conjunto de segmentos que deve ser alocado a cada membro da equipe de serviço e o roteiro a ser percorrido. O tempo máximo da jornada de trabalho é de 8 horas.

Na primeira fase de resolução do problema é determinado um roteiro gigante com início e fim no vértice 1. Como neste grafo todos vértices são de grau par cada aresta é atravessada exatamente uma vez, e o ciclo resultante dado pela seqüência 1, 2, 4, 3, 1.

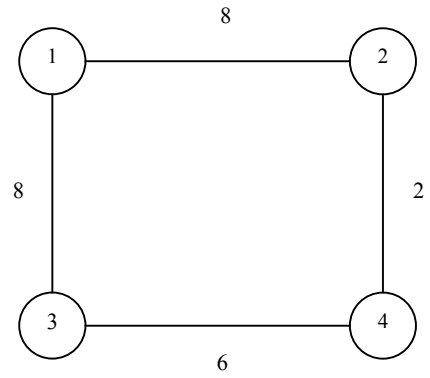


Figura 14 – Rede de atendimento representada por um grafo não orientado valorado

Passa-se a seguir para a segunda fase de resolução do problema, na qual é feito o particionamento do roteiro gigante em uma série de roteiros viáveis. Como neste exemplo todos os segmentos incluídos no roteiro gigante apresentam demanda por serviço, o tempo total de atendimento da rede (C_T) é de 24 horas. A jornada de trabalho individual (n) é de 8 horas, o que conduz a um valor de $k = 24 / 8 = 3$. Devem então ser definidas 3 partições no grafo.

É determinado como ponto de início do percurso da primeira partição o vértice 1. É atravessada, então, a aresta que também incide no vértice 2. Como são necessárias 8 horas para o atendimento deste segmento, fica estabelecida, portanto, a partição SG_1. O processo é repetido a seguir, tendo como ponto de início do segundo roteiro o vértice 2. Como $q_{24} = 2$ e $q_{43} = 6$, a partição SG_2 compreenderá as arestas e_{24} e e_{43}. Por fim, a terceira partição (SG_3) é formada pela aresta incidente nos vértices 3 e 1, onde o tempo de atendimento é de 8 horas. A representação do grafo particionado é feita na figura 15.

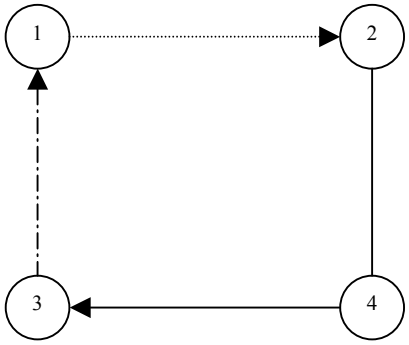


Figura 15 – Grafo não orientado composto de 3 partições

Quando é empregada a estratégia de agrupar primeiro – roteirizar depois, a fase inicial de resolução do problema é compreendida pelo particionamento do grafo que representa a rede de atendimento, e que é feito em função do valor limite da(s) restrição(s) considerada(s). A fase de particionamento é composta de dois procedimentos básicos: um de agregação e outro de melhoria.

No procedimento de agregação, arestas são alocadas as partições (sendo que cada aresta pertence a uma e somente uma partição) até ser atingido o valor limite da restrição considerada no problema. Cada partição é construída a partir de um vértice, escolhido aleatoriamente ou segundo um critério pré-definido.

O procedimento de melhoria, executado a seguir, envolve a trocas de arestas entre partições adjacentes a fim de ser atingido um equilíbrio de demanda ou tempo entre elas. Podem ser adotadas aqui as técnicas citadas no capítulo 2 (item 2.6).

Na segunda fase da estratégia, em cada partição, é determinada a seqüência a ser percorrida através aplicação do algoritmo de roteirização adequado ao tipo da rede com a qual se está trabalhando. Ao contrário da estratégia anterior, existe aqui uma maior flexibilidade no tipo de percurso a ser determinado em cada partição, que poderá ser tanto um caminho ou um ciclo.

A exemplo das observações feitas na estratégia de roteirizar primeiro – agrupar depois, sobre os valores do limite inferior e do limite superior da solução do problema, aqui também são tratados os casos das restrições de tempo máximo para jornada de trabalho e limite de carregamento.

Quando o problema tem como restrição o tempo máximo para a jornada de trabalho, os limites de solução do problema se restringem a um único valor, já que não são conhecidos na fase de particionamento os trechos de percurso ocioso. Assim, o número de partições a serem construídas em G é determinado através da expressão $k = C_T/n$, sendo C_T também neste caso o somatório dos coeficientes q_{ij} dos trechos de percurso efetivo. No caso de restrições de

carregamento valerá a mesma lógica, uma vez que na determinação de C_T são considerados somente os segmentos com $q_{ij} > 0$.

Ainda no caso dos problemas com restrição de tempo máximo para a jornada de trabalho, é possível perceber que o acréscimo dos trechos de percurso ocioso na fase de roteirização poderá acarretar uma violação do tempo limite em algumas partições. Para contornar esta situação pode ser adotado na fase de particionamento um valor para n ligeiramente inferior a aquele que representa o valor real para o limite da jornada de trabalho.

Outro artifício apresentado por Levy e Bodin (1989), consiste na transformação do grafo original em um grafo euleriano onde as arestas incidem aos pares nos vértices. Para isso, pode ser necessário o acréscimo ao grafo original de *arestas fictícias* com coeficiente de custo igual ao tempo de percurso ocioso. Os coeficientes destas arestas acrescentadas a G são consideradas no cálculo de C_T. Após ser terminada a fase de particionamento estas arestas são removidas de G, sendo possível, então, obter partições em que mesmo sendo acrescentados os trechos de percurso ocioso não é violado o valor limite estabelecido.

No quadro 5 são citados os trabalhos em que é tratada a resolução do PRAC através do emprego dos métodos construtivos de duas fases.

No trabalho de Beltrami e Bodin (1974) é feita uma breve análise dos resultados obtidos a partir do emprego destas duas estratégias. Segundo os autores a estratégia de roteirizar primeiro – agrupar depois gera roteiros com menor percurso ocioso se comparado àqueles produzidos pela estratégia de agrupar primeiro – roteirizar depois. Porém, têm a desvantagem de produzir roteiros que se sobrepõem, o que pode gerar dificuldades de ordem administrativa, ao contrário da estratégia de agrupar primeiro – roteirizar que tende a produzir áreas de atendimento com limites mais bem definidos.

Com o objetivo de amenizar estes efeitos negativos, Bodin e Kursh (1978 e 1979) apresentaram um procedimento no qual é aberta a possibilidade de utilização das duas estratégias na resolução de um mesmo problema. Neste caso particular de aplicação dos métodos

construtivos de duas fases, o problema tratado envolvia o serviço de varrição de ruas por equipamentos mecânicos em Nova York.

Quadro 5 - Trabalhos relatando a aplicação dos métodos construtivos de duas fases na resolução do PRAC

Referência	Estratégia utilizada	Aplicação
Stern e Dror (1979)	roteirizar primeiro – agrupar depois	leitura de medidores de energia elétrica
Ulusoy (1985)	roteirizar primeiro – agrupar depois	desenvolvimento teórico
Chapleau *et al* (1984)	agrupar primeiro – roteirizar depois	transporte escolar por ônibus
Levy e Bodin (1989)	agrupar primeiro – roteirizar depois	entrega de correspondência
Bodin *et al.* (1989)	roteirizar primeiro – agrupar depois	coleta de lixo
Bodin e Levy (1991)	agrupar primeiro – roteirizar depois	desenvolvimento teórico
Wunderlich *et al.* (1992)	agrupar primeiro – roteirizar depois	leitura de medidores de gás
Mourão (1997)	roteirizar primeiro – agrupar depois	coleta de lixo
Amberg *et al* (2000)	roteirizar primeiro – agrupar depois	aspersão de sal em vias de tráfego
Muydermans *et al* (2002)	agrupar primeiro – roteirizar depois	aspersão de sal em vias de tráfego

3.4 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Os métodos construtivos de duas fases se caracterizam pela divisão do processo resolutivo em duas fases distintas que são o particionamento da rede de transporte, no qual é determinado o conjunto de segmentos de vias que será designado a cada unidade pertencente a uma frota ou equipe, e a roteirização onde é determinada a seqüência de percurso.

Duas estratégias de resolução compõem o escopo dos métodos construtivos de duas fases: *roteirizar primeiro – agrupar depois* e *agrupar primeiro – roteirizar depois*.

Embora ambas estratégias contemplem uma fase de particionamento e uma fase de roteirização, mesmo que alternadamente, observam-se resultados distintos devido à própria natureza do processo resolutivo.

No capítulo seguinte é feita a proposição de um modelo para resolução do PRAC em redes representadas por grafos não orientados, desenvolvido a partir da estratégia de agrupar primeiro – roteirizar depois.

4 PROPOSIÇÃO DE UM MODELO PARA RESOLUÇÃO DO PROBLEMA DE ROTEIRIZAÇÃO EM ARCOS CAPACITADO

4.1 INTRODUÇÃO

Segundo menção feita no capítulo 1, é bastante significativo o potencial de aplicação dos modelos de roteirização em arcos no planejamento operacional de serviços efetuados por frotas ou equipes de trabalho alocadas ao atendimento de demandas existentes ao longo dos segmentos que compõem um sistema viário. Este fato motivou a introdução do PRAC, cujo modelo matemático leva em consideração restrições de ordem operacional observadas nos serviços enquadrados no escopo dos Problemas de Roteirização em Arcos. Levando em conta ainda o elevado grau de complexidade aí existente, o desenvolvimento de modelos para resolução do PRAC representa um campo de pesquisa bastante promissor na área de otimização.

Apesar disso, nota-se que é ainda reduzido o número de trabalhos publicados em que é dada ênfase à proposição de modelos de resolução para o PRAC. Em termos comparativos, verifica-se um número significativamente maior de trabalhos dessa natureza, dedicados a versão capacitada dos Problemas de Roteirização em Nós.

Considerando esta oportunidade para novos desenvolvimentos, neste capítulo é feita a proposição de um modelo de resolução para o PRAC em redes de transporte representadas por grafos não orientados. No item 4.2 são apresentadas as características do modelo proposto. A seguir, no item 4.3, é feita a apresentação e a descrição do funcionamento dos passos que compõem a seqüência de resolução contida no modelo. A descrição da implementação computacional do modelo proposto é apresentada no item 4.4. Por último, no item 4.5 são feitas as considerações finais deste capítulo.

4.2 CARACTERÍSTICAS DO MODELO PROPOSTO

Devido ao elevado grau de complexidade apresentado pelo PRAC, iniciativas no sentido de resolvê-lo de maneira exata podem se tornar infrutíferas em razão do excessivo tempo de processamento computacional requerido. Isto explica a tendência observada nas referências bibliográficas, do emprego de métodos aproximativos no seu processo de resolução, principalmente quando devem ser tratados problemas com um grande número de variáveis como os que ocorrem em situações de ordem prática.

Em função disso e dos relatos existentes na bibliografia sobre a qualidade das soluções obtidas, também foi adotada como linha de desenvolvimento para o modelo proposto neste trabalho a dos métodos aproximativos. É importante salientar, porém, que este tipo de método se caracteriza pela obtenção de boas soluções (que não são ótimas) em tempos computacionais aceitáveis.

Entre as três categorias em que se dividem os métodos aproximativos aplicados na resolução do PRAC, apresentadas anteriormente no capítulo 2 (item 2.6), o modelo proposto enquadra-se na dos *construtivos de duas fases*, sendo empregada à estratégia de agrupar primeiro – roteirizar depois.

A opção por esta estratégia foi feita em função de seu potencial de aplicação nos casos em que é necessário determinar áreas de atendimento com limites bem definidos e roteiros que não se sobreponham. Foi considerado também o reduzido número de trabalhos em que é relatado o seu emprego na resolução do PRAC.

Uma vez que o desenvolvimento do modelo se deu segundo uma abordagem heurística, tiveram de ser consideradas na definição da estratégia de solução certas características relacionadas ao problema a ser tratado. Estas características são as seguintes:

a) A rede de atendimento é representada por um grafo (ou multigrafo) não orientado;

b) A frota ou equipe de trabalho a ser alocada a rede de atendimento possui características homogêneas (mesma capacidade);

c) O roteiro de cada unidade de trabalho tem início e fim em um mesmo ponto (vértice) localizado em sua área de atendimento;

d) O problema apresenta restrição relacionada ao tempo máximo para jornada de trabalho.

Assim, o modelo tem como objetivo determinar o conjunto de segmentos de via que cada unidade da frota ou equipe deve atender e a seqüência a ser percorrida, de modo que cada roteiro inicie e termine no mesmo vértice, respeitando-se a restrição imposta.

4.3 APRESENTAÇÃO E DESCRIÇÃO DO MODELO

Neste item é feita a apresentação e a descrição do modelo proposto para resolução do Problema de Roteirização em Arcos Capacitado - PRAC. Um protótipo deste modelo foi apresentado por Araújo e Michel (2003).

Em termos gerais, a estrutura do modelo é composta por 5 passos principais. Na figura 16 esta estrutura é apresentada esquematicamente através de um fluxograma.

De acordo com a lógica dos métodos construtivos de duas fases, e em particular da estratégia adotada para desenvolvimento do modelo, é possível identificar a seguinte inserção dos passos:

Primeira fase:

a) Construção das partições;

b) Eliminação das pseudopartições;

c) Melhoria.

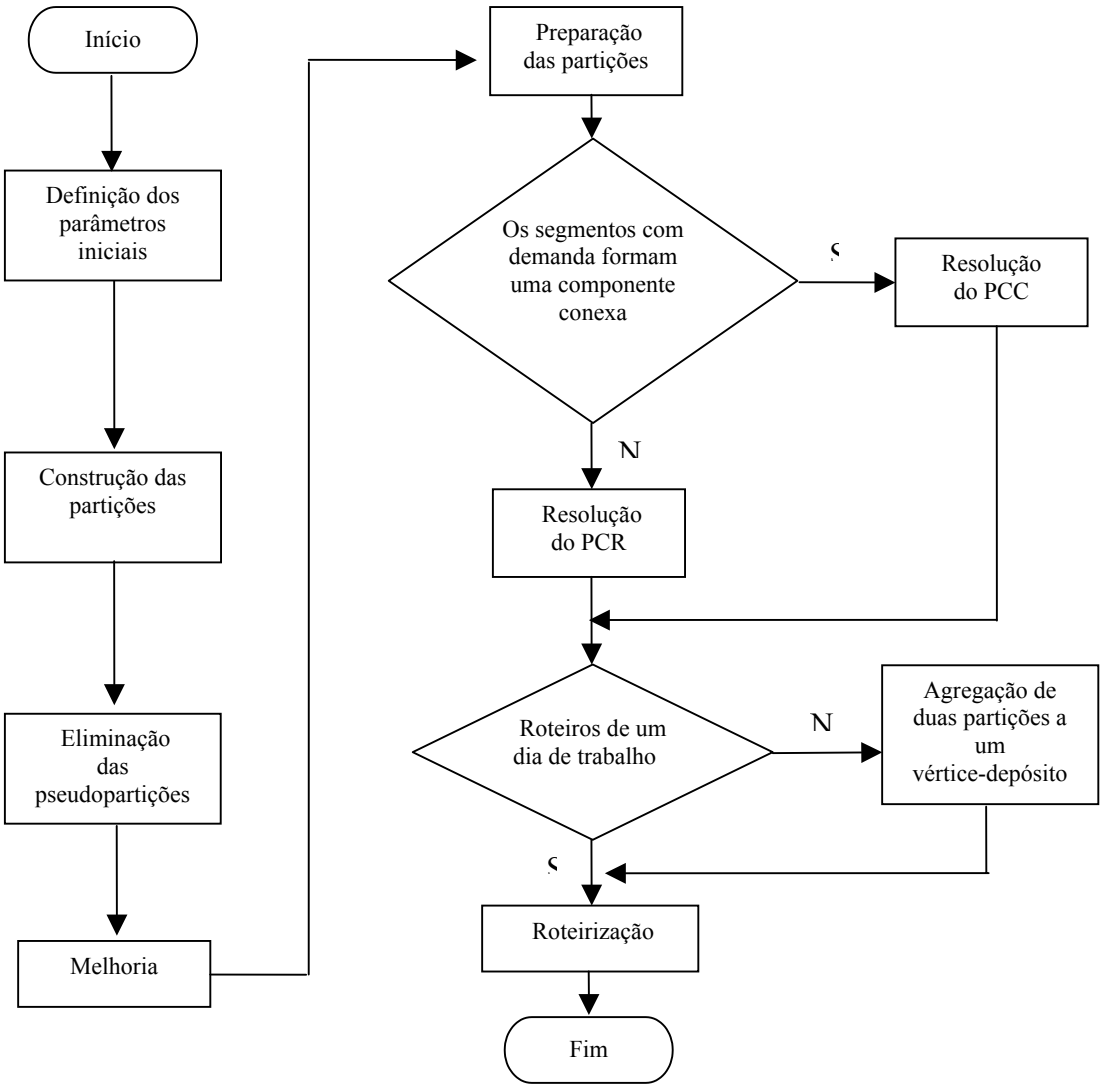


Figura 16 – Fluxograma da estrutura do modelo

Segunda fase:

a) Preparação das partições;

b) Roteirização.

Os novos procedimentos, introduzidos a partir deste modelo, estão compreendidos na primeira fase e envolvem a *construção das partições*, a *eliminação das pseudopartições* e a *melhoria da solução inicial*.

Abaixo são apresentados os passos que compõem o modelo proposto. É feito também após a apresentação de cada passo um comentário sobre sua lógica de funcionamento.

Início

Passo 1. Definição dos parâmetros iniciais.

Passo 1.1. Determine o custo total de atendimento (C_T) da rede através da leitura dos coeficientes de custo c_{ij} das arestas que representam os segmentos de via com demanda por serviço. Vá para o passo 1.2.

Passo 1.2. Informe o tipo de turno a ser considerado e o valor limite (n) para a jornada de trabalho. Vá para o passo 1.3.

Passo 1.3. Se os roteiros a serem determinados em G tiverem a jornada de um dia e o número de partições estimado pelo modelo (k) é um valor fracionário, vá para o passo 1.3.1, senão vá para o passo 2. Se os roteiros a serem determinados em G tiverem jornada de um turno e o número de partições estimado pelo modelo (k) é um valor fracionário e/ou ímpar, vá para o passo 1.3.2, senão vá para o passo 2.

Passo 1.3.1. Informe a opção desejada: arredondar o valor de k para o inteiro superior seguinte diminuindo o valor de n nas demais partições ou arredondar o valor de k para o inteiro inferior seguinte diluindo nas demais partições a parte fracionária excedente.

Passo 1.3.2. Se o valor de k estimado é um número fracionário e/ou ímpar informe a opção desejada: arredondar o valor de k para o inteiro par superior seguinte

diminuindo o valor de *n* nas demais partições ou arredondar o valor de *k* para o inteiro par inferior seguinte diluindo nas demais partições a parte excedente.

Comentário: a execução do modelo inicia-se no passo 1, tendo como primeiro dado de entrada o grafo que representa a rede de atendimento. A seguir são feitos a identificação e o somatório dos coeficientes de custo das arestas (C_T), que apresentam demanda por serviço. É informado então ao usuário o tempo total necessário para o seu atendimento. Na seqüência, o usuário informa ao modelo se deverão ser construídos roteiros com jornada de um dia ou jornada de um turno e o valor limite (*n*) para a jornada de trabalho de uma unidade de trabalho.

Fazendo $k = C_T/n$, o modelo informará ao usuário a quantidade de partições que devem ser construídas em *G*. Porém, esta primeira estimativa pode eventualmente resultar em um valor fracionário no caso da opção por roteiros compreendidos na jornada de um dia e em um valor fracionário e/ou ímpar no caso de roteiros compreendidos em jornada de um turno. Em função disso, no passo 1.3, são apresentadas opções de contorno ao usuário para que este, através de seu julgamento, as avalie e decida pela que achar mais adequada.

Uma vez conhecidos os parâmetros que orientarão a divisão da rede, segue-se para o passo 2 do modelo onde é executado o procedimento de construção das partições.

Passo 2. Construção das partições.

Passo 2.1. Enquanto existirem arestas no grafo (*G*) que não foram alocadas a nenhuma partição (*SG*) vá para o passo 2.2. Senão vá para o passo 3.

Passo 2.2. Identifique o primeiro vértice v_i em *G* com menor grau maior ou igual a 2. Construa um conjunto (tv_p) de vértices candidatos a *vértice-pai* (v_p). Coloque v_i como primeiro elemento de tv_p. Crie uma partição *SG* vazia. Vá para o passo 2.3.

Passo 2.3. Se existirem arestas em G ainda não alocadas a algum SG, tome o primeiro elemento de tv_p e o rotule como v_p (vértice-pai). Coloque v_p em tv_p. Vá para o passo 2.4. Se não há mais elementos em tv_p a partição SG está pronta. Volta ao passo 2.1.

Passo 2.4. Identifique os vértices adjacentes à v_p que são chamados de v_f (vértice-filho). Para cada v_f verifique se a(s) aresta(s) que o liga(m) a v_p pode(m) ser agregada(s) a partição que está sendo construída, através da adição do coeficiente de custo da aresta ao somatório atual da partição (C_{TSG}). Se o novo valor obtido para C_{TSG} for menor ou igual a n remova esta(s) aresta(s) de G e a(s) atribua a SG. O elemento v_f é colocado em tv_p tornando-se assim um v_p. Vá para o passo 2.3.

Comentário: a construção de uma partição inicia em um vértice qualquer localizado no limite externo do grafo, no qual o grau de incidência (que é igual ao número de arestas que incidem no vértice) seja o menor valor maior ou igual a 2. Este elemento é armazenado em um conjunto chamado tv_p a partir do qual são chamados os *vértices-pai* (v_p).

A escolha de um vértice com esta característica objetiva induzir a construção das partições de *fora para dentro* do grafo, evitando assim que após o término deste procedimento possam ocorrer situações em que algumas arestas fiquem isoladas sem terem sido agregadas por alguma partição.

O principio básico de construção de uma partição consiste na agregação das arestas que ligam um *vértice-pai* aos seus sucessores, os *vértices-filho* (v_f). Este procedimento é executado enquanto existir uma aresta entre v_p e v_f ainda não verificada e enquanto a adição do coeficiente de custo da aresta ao somatório atual da partição C_{TSG} for menor ou igual ao valor limite da restrição considerada. Toda vez que uma aresta é adicionada à partição que está sendo construída, o v_f por ela incidido será armazenado em tv_p e se tornará um futuro candidato a v_p.

Se o valor limite da restrição não foi atingido e não existem mais sucessores de v_p para serem verificados, então o primeiro elemento de tv_p ainda não utilizado se torna o próximo v_p. O processo se repete até que não existam mais elementos ainda não utilizados em tv_p ou até que o

somatório C_{TSG} atinja o valor limite da restrição considerada. A construção da partição é encerrada quando uma destas condições ocorrer. A partição construída é então subtraída do grafo original (G), no qual se iniciará o processo de construção de uma nova partição. No momento em que for verificado que cada uma das arestas de G está alocada a uma partição é encerrada a fase de construção, seguindo-se então para o passo 3.

Passo 3. Eliminação das pseudopartições.

> *Passo 3.1.* Se o número de partições construídas no passo 2 for maior do que k, então vá para o passo 3.2. Senão vá para o passo 4.
>
> *Passo 3.2.* Identifique a partição de menor custo, agregando-a a seguir a partição adjacente que apresentar o menor valor de C_{TSG}. Repita até que o número de partições seja igual a k.

Comentário: o procedimento executado no passo 3 tem como finalidade a eliminação das *pseudopartições*. Uma pseudopartição pode ser definida com uma partição construída no passo anterior, na qual o valor de C_{TSG} é significativamente menor do que o valor estabelecido para n. A eliminação de uma pseudopartição é feita através da sua agregação a partição adjacente que apresentar o menor valor de C_{TSG}. Este processo é repetido até que seja atingido o valor de k que representa o número de partições que devem existir em G, conforme estabelecido anteriormente no passo 1. A execução do passo 3 pode resultar em situações onde uma ou mais partições apresentem violação no valor de n. Esta situação é contornada no passo 4, onde é executado o processo de melhoria da solução inicial.

Passo 4. Melhoria.

> *Passo 4.1*. Em cada partição *SG* identifique as arestas que se situam na vizinhança com as demais partições adjacentes. Estes elementos são rotulados como *arestas promissoras* (e_p) para os possíveis movimentos de troca de arestas entre partições vizinhas. Para cada troca possível é associado um valor de ganho. Vá para o passo 4.2.

Passo 4.2. Cálculo do ganho da troca.

Passo 4.2.1. Seja SG_i a partição onde e_p se encontra atualmente e SG_j a partição vizinha potencial recebedora de e_p. Seja C_{TSGi} o custo atual de atendimento de SG_i, C_{TSGj} o custo atual de atendimento de SG_j e c_{ij} o coeficiente de custo associado a e_p. Seja $P_{ij} = |C_{TSGi} - C_{TSGj}| - |(C_{TSGi} - c_{ij}) - (C_{TSGj} + c_{ij})|$ o ganho no custo resultante da troca de e_p de SG_i para SG_j.

Passo 4.2.2. Seja V_i o número de vértices existentes em SG_i e V_j o número de vértices existentes em SG_j antes da troca de e_p, V'_i o número de vértices existentes em SG_i e V'_j o número de vértices em SG_j depois da troca de e_p. Seja então $T_{ij} = V_i + V_j - (V'_i + V'_j)$ o ganho no *fator de forma*.

Passo 4.2.3. Calcule o ganho total da troca $R_{ij} = P_{ij} + 20(T_{ij})$.

Passo 4.3. Faça a troca de maior ganho, caso seja positiva. Repete passo 4.1. Se o maior ganho for negativo, fim.

Comentário: no passo 4 é executado um procedimento de melhoria através das trocas de arestas entre partições adjacentes, tendo como objetivo tornar o mais homogêneo possível o valor de C_{TSG} em cada uma das partições e o seu formato. O procedimento desenvolvido para esta aplicação é classificado como de *busca local*, uma vez que, são definidos elementos específicos que participarão do processo de melhoria (as arestas promissoras – e_p) evitando-se deste modo uma explosão combinatorial.

Neste procedimento todas as arestas situadas no limite de uma partição são potenciais candidatas à troca, sendo denominadas de *arestas promissoras* (e_p). O procedimento é orientado pelo ganho total (R_{ij}) a ser obtido com a possível troca de cada uma das arestas promissoras de uma partição SG_i para uma partição adjacente SG_j. O ganho total compreende duas parcelas: uma relacionada com a melhora no valor de C_{TSG} que é dado por P_{ij}; e outra com o *fator de forma* de SG, que é dado por T_{ij}.

A melhora no valor de C_{TSG} é medida através da verificação de seu valor antes e após a troca de uma aresta. Fica claro que, uma vez executada a troca, quanto mais próximo o valor de C_{TSG} ficar em relação à n, maior o valor do ganho obtido nesta parcela.

Já o fator de forma tem como função acrescentar ao processo de melhoria a busca por um formato de partições o mais uniforme possível. Um formato uniforme aqui significa a mínima ocorrência de *pontes*, ou seja, uma seqüência de arestas isoladas situadas no limite de uma partição que avançam sobre uma partição adjacente. Este critério é avaliado pelo número de vértices existentes em uma partição, sendo que quanto maior o número de vértices maior é a quantidade de pontes. O fator de forma procura minimizar assim, o número de vértices em cada uma das partições. Dada a importância do fator de forma no processo de melhoria, é dado a ele um peso 20 vezes maior do que a avaliação do valor de C_{TSG}. Este peso foi determinado empiricamente através de tentativa e erro, considerando seu impacto na degradação do valor de C_{TSG}.

Este procedimento de melhoria orientado pelo ganho obtido com a troca é repetido até momento em que o valor do ganho passe a ser negativo. A ocorrência de um valor negativo indica que não existem mais possibilidades de ganho, sendo então encerrada a primeira etapa do modelo e iniciada a segunda e última etapa, na qual é determinado o roteiro a ser percorrido em cada uma das partições.

Passo 5. Preparação das partições.

> *Passo 5.1*. Para $SG = 1$ a k verifique se o conjunto dos segmentos com demanda por serviço (E_R) formam uma componente conexa. Se formarem uma componente conexa vá para o passo 5.2 senão vá para o passo 5.3. Se todas as partições estiverem preparadas vá para o passo 5.4.
>
> *Passo 5.2*. Preparação para resolução do Problema do Carteiro Chinês.

Passo 5.2.1. Verifique o grau de incidência (g_i) dos vértices que compõem *SG*. Se todos os vértices tiverem g_i igual a um valor par, então a partição está preparada, senão vá para o passo 5.2.2

Passo 5.2.2. Reúna todos os vértices de grau ímpar de *SG* e construa um grafo completo (R_T) associando a cada par de vértices v_i e $v_j \in R_T$ uma aresta com peso igual ao caminho mais curto que liga v_i a v_j em *SG* definido pelo coeficiente c'_{ij} (tempo de percurso ocioso) das arestas de *SG*. Vá para o passo 5.2.3.

Passo 5.2.3. Determine o emparelhamento perfeito de peso mínimo (E_M) em R_T. Para cada aresta de E_M associe uma nova aresta em *SG* no caminho mínimo que ela representa, de modo que a partição *SG'* resultante tenha todos os seus vértices de grau par.

Passo 5.3. Preparação para resolução do Problema do Carteiro Rural.

Passo 5.3.1. Seja *C* o conjunto que representa como vértices as componentes desconexas de E_R existentes em *SG*, e SG_C o respectivo grafo condensado. Uma aresta e_{xy} de SG_C existe se houver uma aresta $e_{ij} \in SG$ onde $v_i \in C_x$ e $v_j \in C_y$. Defina a distância entre cada par de vértices v_x e $v_y \in SG_C$ como $d(v_x, v_y) = d(C_x, C_y) = \min_{i,j}$ {caminho mínimo $(v_i, v_j) \setminus v_i \in C_x$ e $v_j \in C_y$} multiplicado por λ, sendo $\lambda = 1{,}1$ [custo $d(v_x, v_y)$] inicialmente. Aplique o algoritmo de árvore de peso mínimo sobre SG_C. Seja $E_{To}(\lambda)$ o conjunto de arestas determinadas a partir da resolução do problema de árvore de peso mínimo.

Passo 5.3.2. Simplifique $E_{To}(\lambda)$ eliminando todas as cópias duplicadas de arestas que estejam em paralelo. Denomine o conjunto de arestas resultante como $E_T(\lambda)$.

Passo 5.3.3. Construa um grafo completo R_T a partir de $SG \cup E_T(\lambda)$ determinando a seguir um emparelhamento perfeito de peso mínimo. Seja $E_M(\lambda)$ o conjunto de

arestas obtidas a partir da resolução do problema de emparelhamento. A solução para o PCR é então dada por $SG(\lambda) = SG \cup E_T(\lambda) \cup E_M(\lambda)$.

Passo 5.3.4. Repita os passos 5.3.1, 5.3.2 e 5.3.3, acrescentado 0,1 ao coeficiente até que se chegue a $\lambda = 1{,}6$ [custo $d(v_x, v_y)$], gerando assim um conjunto de soluções para o PCR em *SG*. Selecione a melhor (de menor custo) entre todas aquelas obtidas a partir desta abordagem.

Passo 5.4. Agregação de duas partições a um vértice-depósito.

Passo 5.4.1. Se as partições compreenderem roteiros com jornada de um dia de trabalho vá para o passo 6, senão vá para o passo 5.4.2.

Passo 5.4.2. Construa um grafo condensado completo C_c onde cada partição (*SG*) corresponde a um vértice. Se duas partições adjacentes SG_i e SG_j compartilham um vértice-depósito é criada uma aresta que as liga em C_c com peso igual a 1; se não compartilham um vértice-depósito é criada uma aresta com peso igual a 10; e se SG_i e SG_j não são adjacentes é criada uma aresta com peso 100.000. Vá para o passo 5.4.3.

Passo 5.4.3. Resolva um problema de emparelhamento perfeito de peso mínimo em C_c, cujo resultado corresponderá a duas partições adjacentes (representadas por vértices em C_c) unidas pelo vértice-depósito. Identifique os pares de partições formadas e agregue a cada um destes pares um vértice-depósito que esteja localizado entre elas. Caso não haja um vértice-depósito tome um vértice qualquer. Vá para o passo 6.

Comentário: o passo 5 compreende um conjunto de procedimentos que objetivam preparar cada uma das partições para a execução do passo 6 (roteirização), onde cada uma das partições deverá corresponder a um grafo euleriano. No caso de roteiros de um turno, deve ser feita ainda a agregação de duas partições adjacentes a um vértice-depósito.

Primeiramente, em cada partição é definida a subclasse do Problema de Roteirização em Arcos que aí ocorre, que poderá ser o PCC ou o PCR. Esta definição, feita no passo 5.1, segue os mesmos princípios vistos anteriormente no capítulo 2, ou seja, quando os segmentos com demanda por serviço formam uma componente conexa, é caracterizada a ocorrência do PCC e quando os segmentos com demanda por serviço formam componentes desconexas, tem-se a ocorrência do PCR.

A preparação para a resolução do PCC é feita no passo 5.2. Caso a partição em análise não tenha todos os seus vértices de grau par, deverá ser transformada em um grafo aumentado *SG'*. Na construção do grafo completo R_T, a definição dos caminhos mínimos entre cada um dos vértices (que representam as arestas de R_T) é feita com o emprego do algoritmo de *Dijkstra* (anexo A), que possui complexidade da ordem de O ($|V|^2$). Para a resolução do problema de emparelhamento perfeito de peso mínimo é utilizado o algoritmo de *Gabow* (anexo B), cuja complexidade é de $O(|V|^3)$.

Já no caso da preparação para a resolução do PCR, feita no passo 5.3, é empregada a versão modificada do algoritmo de *Christofides* (apresentada no capítulo 2). Neste passo, assim como no anterior, são utilizados os algoritmos de *Dijkstra* para a determinação de um caminho mínimo entre dois vértices e o algoritmo de *Gabow* para a determinação de um emparelhamento perfeito de peso mínimo. Além destes, também é utilizado o algoritmo de *Kruskal* (anexo C), com complexidade de $O(|E \log V|)$, para a determinação de uma árvore de peso mínimo.

Uma adaptação que teve de ser introduzida neste algoritmo para resolução do PCR foi a adoção da multiplicação, ao invés da soma referida no algoritmo original, da penalidade λ sobre as arestas do grafo condensado SG_c. Em virtude da magnitude dos valores dos coeficientes de custo aí definidos, foram adotados sobre estes, incrementos cumulativos partindo de 10% , crescentes em intervalos de 10% até se atingir o valor de 60%. Esta adaptação foi feita de modo empírico e deveu-se a incapacidade, observada em testes preliminares, que os parâmetros estabelecidos no algoritmo original apresentavam em gerar soluções de menor custo.

No passo 5.4 é executado o procedimento de agregação de duas partições de um turno a um vértice-depósito (v_D). É determinada assim, uma partição que deve ser alocada a uma unidade de trabalho onde a jornada diária compreenderá um roteiro que deverá ser percorrido no turno da manhã, como início e fim em v_D e um roteiro que deverá ser percorrido no turno da tarde, também com início e fim em v_D.

A resolução deste problema é feita através da construção de um grafo condensado completo C_c, no qual cada partição de um turno é transformada em um vértice. Em C_c determinam-se quais pares de partições devem ser agregadas a um v_D através da resolução de um problema de emparelhamento perfeito de peso mínimo, sendo para isso empregado o algoritmo de *Gabow*. Os coeficientes de custo das arestas que ligam os vértices de C_c são determinados em função da existência ou não de um v_D situado entre duas partições adjacentes (custo igual a 1 e a 10, respectivamente) e da não adjacência de duas partições (custo igual a 100.000).

Quando todas as partições estiverem preparadas, segue-se para o último passo do algoritmo que compõem o modelo, onde é feita a determinação da seqüência de percurso em cada uma das partições.

Passo 6. Roteirização

> *Passo 6.1*. A partir do vértice-depósito (v_D) atravesse uma aresta contida em *SG* que lhe seja incidente e que não seja uma *ponte* (ou seja, que a sua supressão torne *SG* desconexo). Apague esta aresta incluindo-a no percurso.
>
> *Passo 6.2.* Vá para a extremidade *j* da aresta apagada e repita o passo 6.1, incluindo as arestas ainda não contidas no percurso até que este se termine em v_D.

Fim

Comentário: no passo 6 é feita a definição da seqüência de percurso em cada uma das partições. Tal percurso corresponderá a um ciclo com início e fim em v_D. Foi utilizado aqui o algoritmo de

Fleury, cuja complexidade é dada por $O(|V|)$. Feito isso, é encerrado o processo resolutivo compreendido no modelo.

No próximo item é feita a descrição da implementação computacional adotada para o modelo.

4.4 IMPLEMENTAÇÃO COMPUTACIONAL DO MODELO

A implementação computacional representa um aspecto de fundamental importância no processo de resolução dos problemas de roteirização, independentemente da classe a qual pertencem. Isto se deve a própria natureza dos algoritmos de solução empregados, quase sempre bastante complexos, o que acaba por exigir mecanismos de cálculo automatizados. Soma-se a isto, o grande número de variáveis de decisão, as peculiaridades das restrições tratadas e a complexidade dos sistemas viários que aparecem em problemas de tamanho real.

Deste modo, a implementação computacional de um modelo de roteirização deve ser precedida de uma análise cuidadosa das estruturas de dados e de execução que serão empregadas nos procedimentos resolutivos. Outro aspecto a ser considerado é que a implementação computacional deve favorecer a robustez do modelo.

Na grande maioria dos trabalhos publicados em que são apresentados desenvolvimentos e aplicações de modelos de roteirização, nota-se o emprego de linguagens de programação como forma de implementação computacional dos algoritmos. As linguagens mais utilizadas neste tipo de aplicação são Pascal, Fortran, C, Visual Basic e Java.

A implementação do modelo foi feita em C++, seguindo o paradigma da programação orientada a objeto. Esta escolha foi motivada pela possibilidade do desenvolvimento paralelo de várias rotinas e arquivos que formam o escopo geral do programa.

Para manipulação do registro contendo a estrutura de grafo foi utilizada uma biblioteca de grafos específica, denominada *Graph Template Library* – GTL e disponível em <

http://www.infosun.fmi.uni-passau.de/GTL/ >. Foi utilizada também a ferramenta *Graphlet* para visualização de grafos, disponível em < http://www.infosun.fmi.uni-passau.de/Graphlet/ >.

Por outro lado, a utilização de linguagens de programação apresenta certas limitações no tocante à representação de redes de transporte e seu relacionamento com determinadas informações. Uma prática, já bastante antiga, consiste em codificar a rde através de estruturas simplificadas como matrizes e listas de adjacência, o que pode restringir a representação de suas características físicas.

Atualmente, esta limitação pode ser contornada com o uso combinado das linguagens de programação e dos Sistemas de Informações Geográficas - SIG.

Os SIG´s são sistemas que visam à coleta, o armazenamento, a manipulação, a análise e a apresentação de informações sobre entidades de expressão espacial, isto é, entes cuja localização, forma, posição e conectividade são relevantes. Esta tecnologia pode trazer enormes benefícios ao processo de resolução dos modelos de roteirização, devido a sua capacidade de manipular a informação espacial (georeferenciada) de maneira precisa, rápida e sofisticada.

Um aspecto bastante importante dos dados tratados em um SIG é a natureza dual da informação, uma vez que um dado espacial ou um dado geográfico possui uma localização expressa como coordenadas de um mapa e atributos descritivos representados num banco de dados convencional. Deste modo, o emprego de um SIG possibilita que sejam disponibilizadas informações gráficas (mapas, desenhos, imagens e figuras em geral) e tabulares (atributos relativos aos elementos gráficos) mutuamente relacionadas em um único ambiente.

Entretanto, a manutenção e atualização de uma base de dados contendo as informações de uma rede viária em um SIG são particularmente críticas, principalmente em cidades maiores, nas quais há mudanças freqüentes na ocupação do solo e na circulação de veículos e pedestres. Deve-se assumir assim, o ônus de manter o banco de dados constantemente atualizado, a fim de que não sejam comprometidos os resultados produzidos pelo modelo de roteirização.

Visando agregar ao processo de implementação computacional do modelo proposto as vantagens da utilização dos bancos de dados georeferenciados fornecidos pelos SIG´s, foi empregado o aplicativo denominado Maptitude.

A estrutura de dados empregada na implementação do modelo, a estrutura de arquivos do programa principal apresentada e a listagem do programa principal (main) são apresentadas, respectivamente nos Apêndices A, B e C.

4.5 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Neste capítulo foi feita a proposição de um modelo para resolução do PRAC em redes de transporte representadas por grafos não orientados. Dentre as três categorias que compõem os métodos aproximativos para resolução do PRAC, o modelo proposto enquadra-se na dos *construtivos de duas fases*, tendo sido empregada a estratégia de *agrupar primeiro – roteirizar depois*. Dada a natureza heurística do modelo proposto, seu potencial está direcionado para a obtenção de boas soluções em tempos de processamento computacionais aceitáveis.

Na primeira fase da estratégia de solução, de particionamento, estão compreendidos os novos procedimentos, introduzidos a partir do modelo que são a *construção das partições* e a *melhoria da solução inicial*. Na segunda fase, de determinação da seqüência de percurso em cada uma das partições, são empregados algoritmos desenvolvidos por outros autores e apresentados na bibliografia dos Problemas de Roteirização em Arcos.

Para a implementação computacional do modelo foi empregada a linguagem de programação C++ orientada a objetos, que executa o processo resolutivo contido no modelo; um aplicativo SIG para armazenamento dos dados necessários; uma biblioteca para registro da estrutura de grafo; e um visualizador de grafos.

No próximo capítulo, é apresentada a aplicação do modelo proposto em um estudo de caso real.

5 APLICAÇÃO DO MODELO

5.1 INTRODUÇÃO

Uma aplicação do modelo proposto neste trabalho para resolução do PRAC foi realizada através de um estudo de caso envolvendo o planejamento dos itinerários para atendimento das demandas pelos serviços de leitura de medidores de energia elétrica e entrega de contas existentes ao longo dos segmentos de via em uma rede viária urbana. O objetivo nesta aplicação foi o de determinar o conjunto dos segmentos que iriam formar cada itinerário e a respectiva seqüência em que tais segmentos deveriam ser percorridos.

O item 5.2 apresenta as características físicas e operacionais do ambiente de aplicação do modelo. No item 5.3 é feita a descrição do trabalho de coleta dos dados necessários à execução do modelo. O item 5.4 apresenta o processo de execução do modelo nesta aplicação. A análise dos resultados obtidos e a sua comparação com a prática atualmente adotada são feitas no item 5.5. Por último, no item 5.6, são feitas as considerações finais deste capítulo.

5.2 CARACTERÍSTICAS DO AMBIENTE DE APLICAÇÃO DO MODELO

A aplicação do modelo tomou por base os serviços de leitura de medidores de energia elétrica e entrega de contas que são efetuados em parte da rede viária urbana de uma cidade do interior gaúcho, a qual em um determinado dia de cada mês, uma equipe de trabalho é alocada para o atendimento das demandas. A área utilizada neste estudo de caso é formada um conjunto de itinerários, sendo cada um deles alocado a um leiturista e a um entregador de contas. Em termos matemáticos estes serviços são descritos através do Problema de Roteirização em Arcos Capacitado.

A rotina operacional dos membros da equipe de trabalho tem início em uma base operacional. À partir desta base operacional, cada leiturista e cada entregador de contas se desloca até um ponto qualquer situado dentro ou próximo ao seu itinerário, para então iniciar a

execução das tarefas. Este deslocamento pode ser feito a pé (no caso de itinerários situados junto à base operacional), de ônibus ou com carros e motos da empresa. Uma vez concluídas as tarefas cada membro da equipe deve retornar a base operacional aonde é feito o encerramento do trabalho. Nestes serviços, o tempo máximo de jornada de trabalho em campo de cada membro da equipe deve se situar próximo aos 480 minutos (8 horas).

Na rede de atendimento todos os segmentos que apresentam demanda por serviço devem ser percorridos pelo menos uma vez, sem obrigatoriedade de sentido, uma vez que o deslocamento é feito através do passeio público. Os segmentos que apresentam grande concentração de demanda nas duas faces de quadra podem ser percorridos em duas passadas, ou seja, uma face de quadra sendo atendida de cada vez. Os demais são atendidos em uma única passada, em um movimento de *zigue-zague*.

A prática adotada pela empresa executora destes serviços para determinação do conjunto de vias (itinerário) que cada membro da equipe deve atender consiste de um método empírico, baseado na experiência de um funcionário responsável por esta atividade de planejamento. Não existe definição sobre a seqüência de percurso dos segmentos, sendo que cada leiturista e cada entregador de contas opta pela seqüência que lhe for mais conveniente.

5.3 COLETA DE DADOS

A execução do modelo nesta aplicação exigiu a realização de uma etapa prévia de coleta de dados compreendendo o levantamento e a tabulação de um conjunto de informações relacionadas às características físicas da rede de atendimento e aos coeficientes de custo associados a cada segmento de via.

A manipulação da rede viária foi feita em um aplicativo SIG, a partir da representação gráfica existente na forma de mapa digitalizado. Primeiramente algumas correções (acréscimo e eliminação de segmentos) tiveram de ser feitas em função de terem sido observadas algumas discrepâncias entre a rede viária real e aquela representada no mapa digitalizado. A representação da rede construída no SIG é mostrada na figura 17.

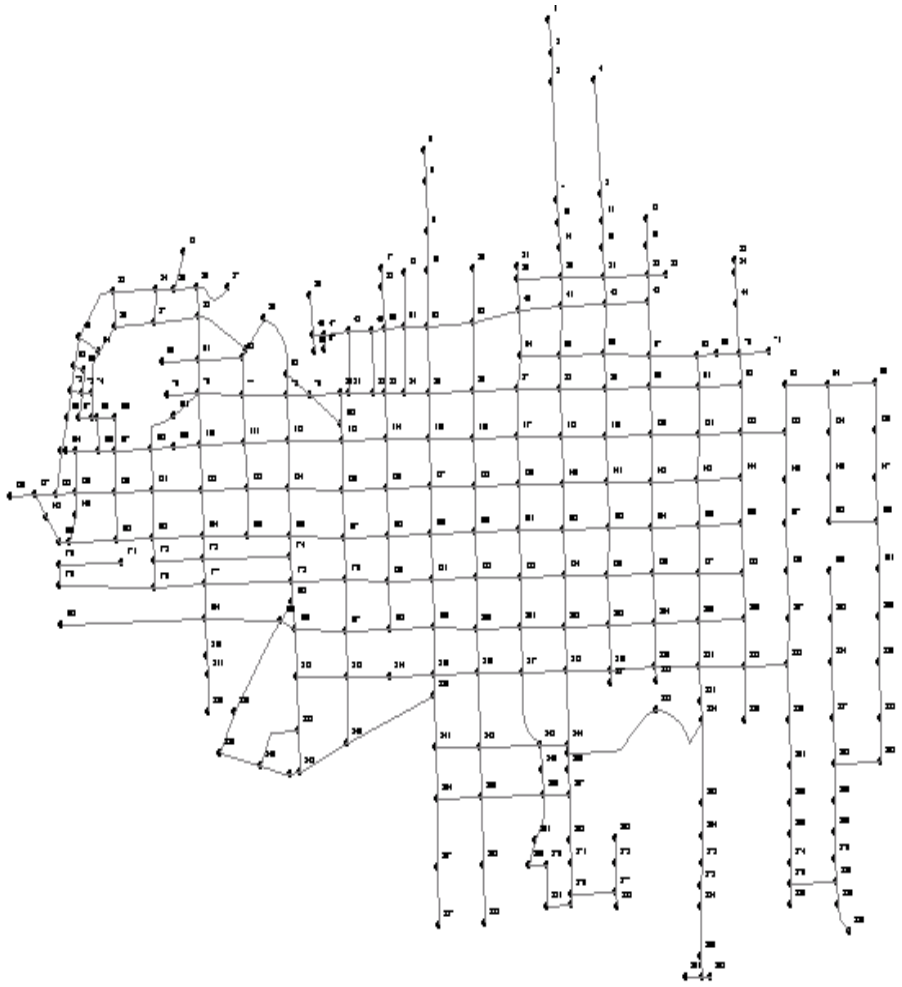


Figura 17 – Representação da rede de atendimento no aplicativo SIG

A rede construída no aplicativo SIG era composta por duas entidades (onde cada uma correspondia a um *layer*): *vértice* e *aresta*. A estas duas entidades foram associados *atributos*, apresentados no quadro 6.

A definição do tempo de percurso efetivo (com trabalho) e do tempo percurso ocioso (sem trabalho) foi feita a partir de listagens fornecidas pela empresa que continham o instante em que cada imóvel era atendido em uma ocasião anterior. Nestas listagens, porém, os instantes de atendimento dos imóveis apareciam de maneira corrida em função do nome do logradouro, sem a identificação, portanto dos segmentos que aí estivessem contidos. Em função disso, teve de ser

realizada uma pesquisa de campo para identificar os números dos imóveis que se localizavam junto às esquinas.

Quadro 6 – Atributos associados às entidades vértice e aresta no aplicativo SIG

Entidade	**Atributo**
Vértice	latitude e longitude da intersecção que o vértice representa
	número do vértice
	rótulo que assume o valor 1 se o pode ser utilizado com vértice-depósito e 0 caso contrário
Aresta	nome do logradouro ao qual pertence o segmento
	comprimento do segmento (em km)
	vértice v_i em que a aresta incide
	vértice v_j em que a aresta incide
	número de identificação da aresta
	rótulo que assume o valor 1 se o segmento demanda serviço e 0 caso contrário
	número de porta do imóvel localizado junto a esquina *i* do lado par do segmento
	número de porta do imóvel localizado junto a esquina *j* do lado par do segmento
	número de porta do imóvel localizado junto a esquina *i* do lado ímpar do segmento
	número de porta do imóvel localizado junto a esquina *j* do lado ímpar do segmento
	número de imóveis que demandavam serviço no lado par do segmento
	número de imóveis que demandavam serviço no lado ímpar do segmento
	rótulo que assume o valor 1 se podem ser atendidas em uma única passada (em *zigue-zague*) as duas fases de quadra compreendidas no segmento e 2 se cada face de quadra é atendida em separado da outra
	tempo gasto no atendimento de todas as demandas existentes no lado par do segmento. Se o segmento é de uma passada este campo não tem valor definido
	tempo gasto no atendimento de todas as demandas existentes no lado impar do segmento. Se o segmento é de uma passada este campo não tem valor definido
	tempo gasto no atendimento de todas as demandas existentes nas duas faces de quadra compreendidas no segmento. Se o segmento requer atendimento em separado de cada face de quadra este campo não tem valor definido
	tempo de percurso ocioso (sem trabalho) entre os dois extremos do segmento

A partir destas informações foi possível identificar nas listagens a quase totalidade dos segmentos existentes na rede viária compreendida na área de estudo. Pode-se então, subtraindo o instante de atendimento da extremidade *i* do instante de atendimento da extremidade *j* definir o tempo gasto no atendimento das demandas existentes em um segmento (tempo efetivo de percurso).

Em um número reduzido de casos, não foi possível identificar nas listagens certos segmentos, devido à divergência entre os dados cadastrais dos imóveis e os dados apurados em campo. Com isto, o tempo efetivo de percurso teve de ser estimado em função do número de imóveis existentes no segmento; do tempo médio de atendimento de cada imóvel, apurado através de medição em campo como sendo de 15 segundos em média; e do tempo de deslocamento.

O tempo de deslocamento em cada segmento foi apurado em função de seu comprimento e da velocidade média desenvolvida pelos leituristas e entregadores quando em percurso ocioso, também apurada através de medição em campo e cujo valor médio é de 3 km/h.

Nesta aplicação do modelo foram considerados também os *vértices-depósito* como pontos de início e fim dos roteiros em cada itinerário. Os vértices-depósito foram definidos antecipadamente em pesquisa de campo e acrescentados aos dados de entrada do modelo. Estes pontos correspondem a facilidades existentes ao longo da rede viária, como locais para alimentação e higiene e pontos de embarque e desembarque do sistema de transporte coletivo por ônibus.

Para execução do modelo, cujos passos de resolução haviam sido implementados em linguagem C++, teve de ser feita à conversão rede construída no aplicativo SIG em um multigrafo não orientado (figura 18), composto por 293 vértices e 622 arestas. Pode-se notar na figura 18 que os segmentos que demandam atendimento em duas passadas são representados por duas arestas incidindo no mesmo par de vértices.

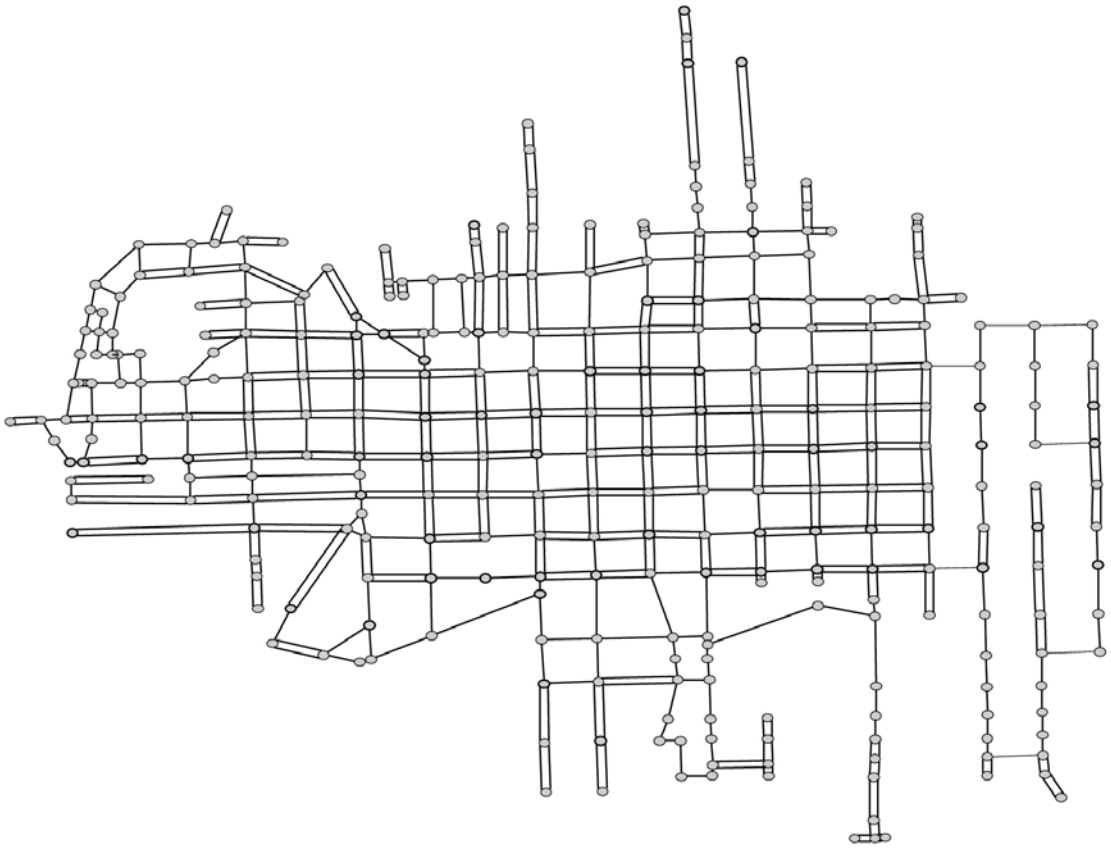


Figura 18 – Representação da rede de atendimento através de um multigrafo não orientado

Uma vez feita à determinação dos parâmetros passou-se a etapa seguinte, de execução do modelo.

5.4 EXECUÇÃO DO MODELO

Para realizar a execução do modelo nesta aplicação foi utilizado um micro-computador tipo *desk-top*, equipado com um processador Pentium IV, *hard disk* de 1,6 GHz e 128 Mb de memória.

A execução do modelo compreendeu a construção de dois cenários, sendo que no primeiro optou-se pela configuração dos itinerários formados por roteiros com jornada de um dia e no segundo pela configuração dos itinerários sendo formados por dois roteiros com jornada de um turno cada.

Para execução do primeiro cenário foi adotado um tempo de 480 minutos (8 horas) como valor de referência para a jornada de trabalho em cada uma das partições. Como a área de atendimento se situava junto à base operacional, o tempo de deslocamento dos leituristas e entregadores de conta até o local de início dos roteiros é bastante reduzido, tendo sido então desconsiderado no cômputo da jornada de trabalho. Em outras aplicações, onde o tempo de deslocamento até a área de atendimento é significativo, este tempo pode ser descontado do valor da jornada de trabalho informado ao modelo.

O tempo total informado pelo modelo para o atendimento das demandas na rede em estudo foi de 6646 minutos. Este valor corresponde ao tempo efetivo (com trabalho), sem considerar o tempo com percurso ocioso. Como o valor de *k* (número de partições) estimado pelo modelo resultou em um valor fracionário (13,8458), duas opções foram apresentadas: o arredondamento para o inteiro imediato inferior, resultando em 13 partições com média de 511,23 minutos, e; o arredondamento para o inteiro imediato superior, tendo-se então 14 partições, com média de 474,714 minutos. A fim de ser mantido o patamar da restrição considerada, foi adotada a segunda opção.

Na seqüência foi executado o passo 2 (construção das partições); o passo 3 (eliminação das pseudopartições); e o passo 4 (melhoria), todos eles compreendidos na fase I do modelo.

Partiu-se a seguir para execução da fase II do modelo, compreendendo o passo 5 (preparação das partições) e o passo 6 (roteirização). Esta fase teve início com a análise da condição de conexidade dos segmentos com demanda por serviço em cada uma das partições. Feito isso, cada partição foi preparada de acordo com a subclasse do Problema de Roteirização em Arcos aí representada (PCC ou PCR) para que, em seguida, pudesse ser definida a seqüência de percurso. Foram designados como ponto de início e fim dos roteiros os vértices-depósito.

O tempo total requerido para a execução do modelo considerando este primeiro cenário foi de 10 segundos. No apêndice D é mostrada a tela de saída completa da execução do modelo para este cenário.

Para a execução do segundo cenário foram adotados roteiros com jornada de um turno, sendo o tempo estipulado em 240 minutos (4 horas). Assim como no caso do cenário anterior, o tempo efetivo requerido para o atendimento da rede foi de 6646 minutos.

O valor de *k*, primeiramente estimado foi 27,6917 partições. As opções fornecidas pelo modelo foram: o arredondamento para o valor imediato inferior par, resultando em 26 partições com média de 255,615 minutos, e; o arredondamento para o valor imediato superior par, resultando em 28 partições com média de 237,357 minutos. Objetivando manter o patamar da restrição considerada, optou-se pelo seu arredondamento para o valor inteiro par superior, correspondendo a 28 roteiros.

Como estes roteiros foram agregados aos pares, o resultado final correspondeu a 14 partições. Pôde-se assim, estabelecer uma comparação entre os resultados obtidos nos dois cenários de execução do modelo.

Para a execução da fase I do modelo neste segundo cenário foram utilizados os mesmos procedimentos do cenário anterior. Na fase II, porém, além da preparação das partições e da roteirização, teve ser executado o procedimento de agregação de pares de partições (onde cada partição correspondia a um roteiro de um turno) aos vértices-depósito.

Neste segundo cenário, o tempo total requerido para a execução do modelo foi de 15 segundos. Este aumento no tempo de processamento em relação aquele demandado na execução cenário anterior se deve ao número maior de iterações que foram necessárias no procedimento de melhoria. No apêndice D é mostrada a tela de saída completa da execução deste cenário.

No item a seguir é feita a análise dos resultados obtidos com esta aplicação do modelo.

5.5 ANÁLISE DOS RESULTADOS OBTIDOS COM A APLICAÇÃO DO MODELO

Uma vez concluída a execução do modelo, pôde ser feita uma análise do seu desempenho e uma comparação entre os resultados obtidos e a prática atual adotada pela empresa executora

dos serviços para planejamento dos itinerários. Foram considerados aqui os resultados obtidos nos dois cenários de execução do modelo.

Para a análise do desempenho do modelo em cada um dos cenários, foram tomados por base os resultados obtidos na fase I, antes e após a execução do procedimento de melhoria, e no término da fase II. Como o critério de ganho utilizado no processo de melhoria da solução inicial compreende uma parcela relacionada com o custo e outra relacionada com o fator de forma, foram adotados como parâmetros de análise em cada partição o custo de atendimento e o número de vértices aí existente. Foram considerados também os valores totais, as médias, os desvios padrões e os coeficientes de variação observados.

A tabela 1 apresenta um relatório contendo os parâmetros de análise, referentes a execução do modelo no primeiro cenário.

Pode-se observar na tabela 1, que os valores obtidos apresentam variação de uma partição para outra. Tal variação se deve a natureza combinatorial do PRAC, onde as arestas correspondem a variáveis inteiras.

Na fase I, antes de ser executada a melhoria da solução inicial, o maior tempo observado para atendimento de uma partição era de 579 minutos e o menor tempo era de 253,2 minutos. O desvio padrão neste momento ficava em 69,86 minutos e o coeficiente de variação em 14,72%. Em relação ao número de vértices, o maior valor observado foi de 50, enquanto o menor era de 12. O desvio padrão era de 12,9 vértices e o coeficiente de variação de 49,8%. Deve-se ressaltar, porém, que o número de vértices depende diretamente do tamanho da partição.

Com a execução do procedimento de melhoria, o maior tempo observado passou a ser de 483,1 minutos, enquanto o menor tempo ficou em 469,9 minutos. Com isso, o desvio padrão reduziu-se para 6,715 minutos e o coeficiente de variação para 1,41%.

Tabela 1 – Resultados obtidos com a execução do modelo no primeiro cenário

	Fase I				**Fase II**
	Antes da melhoria		**Após a melhoria**		**Após a preparação da partição**
Partição	**Tempo para atendimento da partição (minutos)**	**Número de vértices**	**Tempo para atendimento da partição (minutos)**	**Número de vértices**	**Tempo para atendimento da partição (minutos)**
1	500,4	38	483,1	36	493,1
2	480,1	50	480,1	50	490,2
3	487,5	18	477,3	16	485,7
4	485,9	20	477,2	19	481,2
5	474,2	16	476,4	17	478,2
6	513,7	38	477,1	34	488,7
7	473,2	12	481,2	13	484,8
8	472,6	12	469,9	13	478,3
9	474,5	17	473,2	15	480
10	474,7	29	455,3	28	472,3
11	470,9	13	471,6	16	476,2
12	506,1	23	475,7	17	481,1
13	579	46	476,1	42	493,5
14	253,2	31	471,8	45	472,5
Total	**6646**	**363**	**6646**	**361**	**6756,9**
Média	**474,71**	**25,9**	**474,71**	**25,8**	**482,64**
Desvio	**69,86**	**12,9**	**6,715**	**13,1**	**7,026**
Coef. de variação	**14,72%**	**49,8%**	**1,41%**	**50,78%**	**1,46%**

Já o número de vértices, que indica uma melhoria no formato das partições apresentou uma redução de valor em 8 partições, e em uma o valor foi mantido constante. A exceção da partição 14, onde o aumento no número de vértices foi ocasionado pela agregação de novas arestas (em função do baixo tempo de atendimento), os demais incrementos foram pouco significativos. Com isso, o aumento verificado no valor do desvio padrão e no valor do coeficiente de variação foram bastante reduzidos. Deve-se observar também o aumento no número total de vértices constante na tabela em relação ao número de vértice do grafo original ocorre por que vértices que são incididos por arestas de mais de uma partição são contados uma vez por partição.

Como na fase II do modelo são acrescentados os trechos de percurso ocioso, determinados na resolução do PCC e do PCR, houve um incremento nos valores dos tempos, passando a ser de

493,5 minutos o maior tempo e 472,3 minutos o menor tempo. Com isso, houve também uma pequena elevação no valor do desvio padrão que passou a ser de 7,026 minutos. O coeficiente de variação ficou em 1,46%.

O apêndice E apresenta a visualização gráfica e a seqüência de percurso para cada uma das partições determinadas neste primeiro cenário.

Na tabela 2 são apresentados os valores obtidos para os parâmetros de análise do segundo cenário antes de ser feita a agregação dos pares de partições aos vértices-depósito.

Antes de ser executado o procedimento de melhoria, na fase I, o maior tempo observado foi de 376,6 minutos, enquanto o menor tempo ficou em 109,2 minutos. Neste momento, o desvio ficou em 47,88 minutos e o coeficiente de variação em 20,17%.

Em relação ao número de vértices por partição, o maior valor ficou em 36, enquanto o menor ficou em 6. O desvio padrão foi de 9 vértices e o coeficiente de variação em 61,64%.

Após a execução da melhoria, o maior tempo ficou em 280,6 minutos e o menor tempo em 220,5 minutos. O desvio padrão foi reduzido para 10,6 minutos, sendo o coeficente de variação de 4,46%.

Em 14 partições houve redução no número de vértices e em 4 partições o número ficou constante. Nas 10 partições em que houve aumento do número de vértices, somente em um caso (partição 28) houve um aumento significativo e que se deveu ao aumento de tamanho dado o baixo valor de tempo verificado antes da execução da melhoria. Houve também queda no valor do desvio padrão para 7,99 vértices e do coeficiente de variação para 56,66%.

Tabela 2 – Resultados obtidos com a execução do modelo no segundo cenário

	Fase I				**Fase II**
	Antes da melhoria		**Após a melhoria**		**Após a preparação da partição**
Partição	**Tempo para atendimento da partição (minutos)**	**Número de vértices**	**Tempo para atendimento da partição (minutos)**	**Número de vértices**	**Tempo para atendimento da partição (minutos)**
1	237,3	24	237,9	25	243,7
2	236,7	36	231,9	38	246,9
3	255,8	12	234,7	11	237,9
4	237,3	8	234,6	7	237,2
5	230,5	16	236	12	243,8
6	237,1	13	236,3	13	245,7
7	321,9	25	239,6	17	246,6
8	294,3	7	280,6	6	280,6
9	234,4	14	235,9	13	238,7
10	214,6	6	234,9	8	234,9
11	263,8	16	237,6	15	252
12	230,8	10	242,1	8	242,1
13	233,5	6	239,6	7	242,6
14	235,9	9	237,7	8	239,5
15	190,7	14	236,5	15	242,3
16	237,8	16	245,6	17	251,4
17	272,3	29	229,9	27	249,1
18	225,3	6	230	5	231,6
19	235,7	11	252,8	10	257,2
20	228,3	7	231,5	7	231,5
21	236,6	8	230,6	10	232
22	234,6	8	237,8	8	237,8
23	376,6	34	243,4	25	257
24	232	8	231,5	10	233,3
25	229,8	9	225	9	225
26	135,9	12	241,4	17	247
27	237,3	30	220,5	25	238,1
28	109,2	11	230,1	22	244,7
Total	**6646**	**408**	**6646**	**395**	**6810,2**
Média	**237,36**	**14,6**	**237,36**	**14,1**	**243,22**
Desvio	**47,88**	**9**	**10,6**	**7,99**	**10,65**
Coef. de variação	**20,17%**	**61,64%**	**4,46%**	**56,66%**	**4,38%**

Com a execução da segunda fase, e a conseqüente adição dos trechos de percurso ocioso, o maior tempo continuou em 280,6 minutos e o menor tempo passou a ser de 225 minutos. O desvio padrão do tempo teve um pequeno aumento, ficando em 10,65 minutos. Já o coeficiente de variação reduziu-se para 4,38%.

É importante ressaltar que a opção por roteiros com jornada de um turno apresenta valores maiores de desvio padrão e coeficiente de variação do que aqueles observados na opção de jornada de um dia em razão de que o tamanho mais reduzido das partições torna mais sensível a variação do resultado do tempo quando é feita a troca de arestas. Além disso, nas partições de jornada de um turno que apresentam valores elevados para o tempo de atendimento podem ser o resultado da existência de segmentos com significativo tempo de atendimento.

Na tabela 3 são apresentados os resultados obtidos para o tempo de atendimento após a agregação de pares de partições aos vértices-depósito. Nesta situação, o maior tempo foi de 526,3 minutos, enquanto o menor tempo ficou em 463,5 minutos. O desvio padrão foi de 16,776 minutos e o coeficiente de variação de 3,45%.

Tabela 3 – Resultados obtidos para o tempo de atendimento com a agregação de pares de partições aos vértices-depósito

Pares de partições agregadas	**Tempo (minutos)**
1-15	486
2-5	490,7
3-4	475,1
6-8	526,3
7-11	498,6
9-10	473,6
12-13	484,7
14-19	496,7
16-23	508,4
17-25	474,1
18-22	469,4
20-21	463,5
24-26	480,3
27-28	482,8
Total	**6810,2**
Média	**486,44**
Desvio	**16,776**
Coef. de variação	**3,45%**

Nota-se neste caso, uma maior variação no tempo de atendimento das partições em relação às partições com roteiros de jornada de um dia. Isto se deve as limitações impostas pela necessidade de adjacência entre partições com jornada de um turno e de compartilhamento de pelo menos um vértice-depósito, critérios estes utilizados no procedimento de agregação de

partições. Em termos operacionais, porém, esta prática pode ser mais vantajosa do que a adoção de roteiros de um dia, devido a maior flexibilidade para realocação das unidades de trabalho em caso de imprevistos (quebras e condições climáticas adversas, por exemplo).

O apêndice E apresenta a visualização gráfica e a seqüência de percurso para cada uma das partições determinadas neste segundo cenário.

A comparação entre os resultados obtidos com a execução do modelo para estes dois cenários e a prática atual de planejamento adotada pela empresa indica uma redução direta da ordem de 6,67% no número de itinerários necessários ao atendimento da área estudada, uma vez que foram necessários 14 itinerários contra os 15 atualmente existentes. Este valor pode ser considerado bastante significativo, uma vez que a literatura relata ganhos desta natureza da ordem de 2,7% quando são aplicados modelos aproximativos para resolução do PRAC em redes de transporte com características semelhantes a esta (WUNDERLICH *et al*, 1992).

Nos itinerários configurados com roteiros de jornada de um dia são verificadas reduções de 9,53% no tempo total de atendimento da área de estudo e de 3,06% no tempo médio gasto no percurso de um itinerário (tabela 4).

Tabela 4 – Comparação dos resultados obtidos com a execução do modelo no primeiro cenário e a prática atual da empresa

Parâmetro de análise	**Prática atual da empresa**	**Modelo proposto**	**Redução (%)**
Tempo total de atendimento da rede (minutos)	7468,6	6756,9	9,53
Tempo médio gasto por itinerário (minutos)	497,9	482,64	3,06
Número de itinerários	15	14	6,67

Já no caso dos itinerários configurados com dois roteiros de jornada de um turno cada, a redução no tempo total de atendimento da área de estudo foi de 8,82%, enquanto a redução de tempo médio gasto no percurso de um itinerário foi de 2,3% (tabela 5).

Tabela 5 – Comparação dos resultados obtidos com a execução do modelo no segundo cenário e a prática atual da empresa

Parâmetro de análise	Prática atual da Empresa	Modelo proposto	Redução (%)
Tempo total de atendimento da rede (minutos)	7468,6	6810,2	8,82
Tempo médio gasto por itinerário (minutos)	497,9	486,44	2,3
Número de itinerários	15	14	6,67

O gasto anual médio da empresa com estes serviços em toda sua região de atuação, considerando os custos diretos relacionados com mão-de-obra, equipamentos e materiais, é de aproximadamente R$ 2.142.328,44 (tabela 6). Em uma possível aplicação do modelo no planejamento dos itinerários em toda a região de atuação da empresa, mantendo-se o percentual de redução obtido, pode ser estimada uma economia da ordem de R$ 142.893,31 anuais. Porém, este valor pode ser ainda maior se for levada em conta a participação dos custos indiretos.

Por último, deve-se ressaltar que a rede que serviu de base para esta aplicação do modelo compreendia uma área de grande adensamento urbano. Em situações deste tipo, a quantidade de segmentos de percurso ocioso tende a ser menor, uma vez que grande parte destes é atendido em duas passadas. Por outro lado, em áreas com ocupação mais rarefeita o atendimento de grande parte dos segmentos é feito em uma passada, implicando em um aumento na quantidade de percurso ocioso. Deste modo, a manutenção do valor estimado para a economia a ser gerada com a aplicação do modelo em toda região atendida pela empresa apresenta um caráter conservador.

Tabela 6 – Economia estimada com a aplicação do modelo no planejamento dos serviços em toda região atendida pela empresa

Parâmetro de análise	Valor correspondente
Gasto anual médio (R$)	2.142.328,44
Redução na quantidade de itinerários (%)	6,67
Economia estimada (R$)	142.893,31

5.6 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Neste capítulo foi apresentada a aplicação do modelo proposto para resolução do PRAC em um estudo de caso. Nesta aplicação, feita com base nos serviços de leitura de medidores de energia elétrica e entrega de contas executados em parte da rede viária urbana de uma cidade, o objetivo consistia em determinar o conjunto de segmentos de via que cada membro de uma equipe de trabalho deveria atender e a respectiva seqüência de percurso.

Para que fosse feita esta aplicação, foi necessário o desenvolvimento de uma etapa de coleta de dados, que envolveu atividades de campo e de tabulação dos dados obtidos em um aplicativo SIG para posterior utilização na execução do modelo.

A execução do modelo foi feita considerando dois cenários: o primeiro compreendendo a construção de itinerários formados por roteiros de um dia e o segundo itinerários formados por dois roteiros com jornada de um turno cada.

Com base nos resultados gerados nestes dois cenários, foram feitas a análise do desempenho do modelo e a comparação com a situação atualmente existente. Foi determinada também a possível economia a ser obtida com a aplicação do modelo na atividade de planejamento dos itinerários em toda região de atendimento da empresa.

No próximo capítulo são apresentadas as conclusões deste trabalho.

6 CONCLUSÕES E RECOMENDAÇÕES

Os Problemas de Roteirização em Arcos constituem uma classe do Problema de Roteirização de Veículos, no qual as demandas ocorrem de maneira contínua ao longo dos segmentos de via de uma rede de transporte. Seu campo de aplicação é bastante amplo, uma vez que grande parte dos serviços assim modelados apresenta caráter essencial, além de estarem em um processo de contínua mudança devido as crescentes exigências dos clientes e aos avanços tecnológicos, especialmente no campo da telemática.

O Problema de Roteirização em Arcos Capacitado - PRAC, em especial, representa um campo de pesquisa de grande interesse na área de otimização, em razão do número reduzido de trabalhos em que são propostos modelos para sua resolução. Outro aspecto está relacionado a sua característica básica de modelar situações onde a ocorrência de restrições operacionais limita o atendimento de todas as demandas existentes em uma rede de transporte por uma única unidade de trabalho.

A necessidade de definir o conjunto de segmentos de via que cada unidade de trabalho, que pertence a uma frota ou equipe, faz com que além do problema de roteirização deva ser considerado também o Problema de Particionamento de Redes de Transporte.

No caso do PRAC esta situação pode ser descrita através dos métodos construtivos de duas fases, onde são tratados de maneira alternada os processos de roteirização e particionamento. Dada sua natureza heurística, estes métodos caracterizam-se pela obtenção de boas soluções em tempos de processamento computacional aceitáveis.

Este trabalho propôs um modelo para resolução do PRAC quando a rede de atendimento é representada por um grafo não orientado e há restrição ao tempo máximo para a jornada de trabalho. A estratégia adotada no desenvolvimento do modelo foi a de agrupar primeiro –

roteirizar depois que consiste em uma boa opção quando é necessário determinar áreas de atendimento com limites bem definidos, resultando em roteiros com mínima sobreposição.

O modelo proposto é constituído de 5 passos principais, sendo 3 compreendidos na primeira fase e 2 na segunda fase.

Na primeira fase, estão compreendidos os novos procedimentos introduzidos a partir deste modelo e envolvem a construção das partições e a melhoria da solução inicial. Na segunda fase, onde é feita a preparação das partições para a resolução do Problema do Carteiro Chinês – PPC ou do Problema do Carteiro Rural - PCR, são empregados algoritmos desenvolvidos anteriormente por outros autores.

A demonstração do modelo foi feita através de sua aplicação em um estudo de caso real, envolvendo os serviços de leitura de medidores de energia elétrica e entrega de contas, onde foi determinado o conjunto de segmentos de via que cada membro de uma equipe de trabalho tinha de atender e o correspondente percurso. Com os resultados obtidos com a execução de dois cenários, onde foram adotadas, respectivamente, configurações dos roteiros com jornada de trabalho de um dia e jornada de trabalho de um turno, foi possível validar o modelo para gerar boas soluções.

A analise dos resultados obtidos antes e após a execução do procedimento de melhoria na fase I e após a execução da fase II demonstrou a capacidade do modelo de produzir áreas de atendimento com formato uniforme e com pequenas variações no tempo de atendimento, fazendo com que os membros da equipe de trabalho tenham uma jornada de trabalho equilibrada. Ficou comprovada assim, a eficiência dos novos procedimentos de construção das partições e de melhoria da solução inicial, das adaptações que foram feitas no algoritmo de resolução do PCR e do procedimento de agregação de pares de partições a vértices-depósito.

Uma comparação entre os resultados obtidos com a execução do modelo e a prática atual adotada pela empresa mostra uma economia estimada em 6,67%, no tocante a redução do número de itinerários necessários ao atendimento da rede viária da área de estudo. Devido à característica

urbana da rede este percentual pode ser considerado bastante significativo. Deste modo, pôde ser confirmada a hipótese considerada para realização do estudo, apresentada no capítulo 1, de que um modelo desenvolvido para resolução do PRAC a partir de princípios matemáticos, produz melhores soluções do que aquelas geradas pelo método empírico, baseado exclusivamente na experiência prática de um planejador.

As recomendações para futuros trabalhos são as seguintes:

a) Estender os experimentos de aplicação do modelo através do emprego de um gerador aleatório de grafos para testar a robustez do mesmo;

b) Possibilitar a intervenção do usuário a fim de que este defina, através de seu julgamento arestas que devem ser alocadas a partições específicas;

c) Empregar metaheurísticas no processo de melhoria;

d) Estender a abrangência do modelo para as redes de transporte representadas por grafos orientados e mistos;

e) Adaptar ao modelo a técnica desenvolvida por Daganzo (1984) e Newel e Daganzo (1986a; 1986b; 1986c) que busca otimizar o formato das áreas de atendimento em Problemas de Roteirização em Nós, e que em parte já foi aplicado por Carranza e Robusté (1999) no planejamento de áreas de atendimento no serviço de coleta de lixo.

REFERÊNCIAS

AMBERG, A.; DOMSCHKE, W.; VOB, S. Multiple Center Capacitated Arc Routing Problems: A Tabu Search Algorithm Using Capacitated Trees. **European Journal of Operational Research**, v. 124, n. 2, p. 360-376, July 2000.

ARAÚJO, R. R.; MICHEL, F. D. Problemas de Roteirização em Arcos: Características e Métodos de Resolução. In: SIMPÓSIO DE PESQUISA OPERACIONAL DA MARINHA, 4; 2001, Rio de Janeiro. **Anais ...**, Rio de Janeiro, 2001. p. 411-421.

ARAÚJO, R. R.; MICHEL, F. D. Um Modelo de Resolução para o Problema de Roteirização com Restrições de Capacidade. In: CONGRESSO DA ASSOCIAÇÃO NACIONAL DE PESQUISA E ENSINO EM TRANSPORTES, 17; 2003, Rio de Janeiro. **Anais** ..., Rio de Janeiro, 2003. p. 670-681.

BALLOU, R. H. **Business Logistics Management**. New Jersey: Prentice-Hall, 1999. Cap. 6, p. 135-184: Transport Fundamentals.

BELENGUER, J. M.; BENAVENT, E. A Cutting Plane Algorithm for the Capacitated Arc Routing Problem. **Computers and Operations Research**, v. 30, n. 5, p. 705-728, Apr. 2003.

BELTRAMI, E. J.; BODIN, L. Networks and Vehicle Routing for Municipal Waste Collection. **Networks**, v. 4, p. 65-94, 1974.

BEULLENS, P.; MUYLDERMANS, L.; CATTRYSSE, D.; Van OUDHEUSDEN, D. A Guided Local Search Heuristic for the Capacitated Arc Routing Problem. **European Journal of Operations Research**, v. 147, n. 3, p. 629-643, June 2003.

BODIN, L.; FAGIN, G.; WELEBNY, R.; GREENBERG, J. The Design of a Computerized Sanitation Vehicle Routing and Scheduling System for the Town of Oyster Bay, New York. **Computers and Operations Research**, v. 16, n. 1, p. 45-54, 1989.

BODIN, L.; GOLDEN, B.; ASSAD, A.; BALL, M. Routing and Scheduling of Vehicles and Crews. The State of the Art. **Computers and Operations Research**, v. 10, p. 63-211, 1983.

BODIN, L.; KURSH, S. A Computer-Assisted System for the Routing and Scheduling of Street Sweepers. **Operations Research**, v. 26, n. 4, p. 525-537, July-Aug. 1978.

BODIN, L.; KURSH, S. A Detailed Description of a Computer System for the Routing and Scheduling of Street Sweepers. **Computers and Operations Research**, v. 6, n. 4, p. 181-198, 1979.

BODIN, L.; LEVY, L. The Arc Partioning Problem. **European Journal of Operational Research**, v. 53, n. 3, p. 393-401, Aug. 1991.

CABRAL, E. A.; GENDREAU, M.; GHIANI, G.; LAPORTE, G. Solving the Hierarchical Chinese Postman Problem as a Rural Postman Problem. **European Journal of Operational Research**. Jan. 2003. Disponível em < http://www.sciencedirect.com >. Acesso em: 20 de jun. de 2003.

CARRANZA, O.; ROBUSTÉ, F. A Continuos Approximation Model for Vehicle Routing in Solid Waste Management Systems. **Investigacion Operativa**, v. 8, p. 109-153, July-Dec. 1999.

CHAPLEAU, L.; FERLAND, J. A.; LAPALME, G.; ROUSSEAU, J. M. A Parallel Insert Method for the Capacitated Arc Routing Problem. **Operations Research Letters**, v. 3, n. 2, p. 95-99, June 1984.

COOK, C.; SCHOENEFELD, D. A.; WAINWRIGHT, R. L. Finding Rural Postman Tours. In: ACM SYMPOSIUM ON APPLIED COMPUTING, 1998, Atlanta. **Proceedings** ..., New York: ACM Press, 1998. p. 318-326.

CORBERÁN, A.; MARTÍ, R.; ROMERO, A. Heuristics for the Mixed Rural Postman Problem. **Computers and Operations Research**, v. 27, n. 2, p. 183-203, Fev. 2000.

CORBERÁN, A.; MARTÍ, R.; SANCHIS, J. M. A GRASP Heuristic for the Mixed Chinese Postman Problem. **European Journal of Operational Research**, v. 142, n. 1, p. 70-80, Oct. 2002.

CORBERÁN, A.; MARTÍ, R.; MARTÍNEZ, E.; SOLER, D. The Rural Postman Problem on Mixed Graphs with Turn Penalties. **Computers and Operations Research**, v. 29, n. 7, p. 887-903, June 2002.

DAGANZO, C. F. The Distance Traveled to Visit N Points with a Maximum of C Stops per Vehicle: An Analytic Model and an Aplication. **Transportation Science**, v. 18, n. 4, p. 331-350, Nov. 1984.

DASKIN, M. Logistics: Overview of the State of the Art and Perspectives on Future Research. **Transportation Research**, v. 19, part A, p. 383 – 398, Sep-Nov. 1985.

DERIGS, U.; METZ, A. Solving (large scale) Matching Problems Combinatorially. **Mathematical Programming**, v. 50, p. 113-121, 1991.

DROR, M.; LANGEVIN, A. A Generalized Traveling Salesman Problem Approach to the Directed Clustered Rural Postman Problem. **Transportation Science**, v. 31, n. 2, p. 187-192, May 1992.

DROR, M.; STERN, H.; TRUDEAU, P. Postman Tour on a Graph with Precedence Relation on Arcs. **Networks**, v. 17, p. 283-294, 1987.

EDMONDS, J.; JOHNSON, E. Matching, Euler Tours and The Chinese Postman. **Mathematical Programming**, v. 5, p. 88-124, 1973.

EGLESE, R.W. Routeing Winter Gritting Vehicles. **Discrete Applied Mathematics**, v. 48, n. 3, p. 231-244, Feb. 1994.

EISELT, H. A.; GENDREAU, M.; LAPORTE, G. Arc Routing Problems, Part I: The Chinese Postman Problem. **Operations Research**, v. 43, n. 2, p. 231-242, Mar.-Apr. 1995.

EISELT, H. A.; GENDREAU, M.; LAPORTE, G. Arc Routing Problems, Part II: The Rural Postman Problem. **Operations Research**, v. 43, n. 3, p. 399-414, May-June 1995.

FREDERICKSON, G. N. Approximation Algorithms for Some Postman Problems. **Journal of the Association for Computing Machinery**, v. 26, n. 3, p. 538-554, July 1979.

HELSGAUN, K. An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic. **European Journal of Operational Research**, v. 126, n. 1, p. 106-130, Oct 2000.

GABOW, H. N. An Efficient Implementation of Edmonds' Algorithm for Maximum Matching on Graphs. **Journal of the Association for Computing Machinery**, v. 23, n. 2, p. 221-234, Apr. 1976.

GABOW, H.; TARJAN, R. Faster Scaling Algorithms for General Graph-Matching Problems. **Journal of the Association for Computing Machinery**, v. 38, n. 4, p. 815-853, Oct. 1991.

GALIL, Z.; MICALI, S.; GABOW, H. An O(EV log V) Algorithm for Finding a Maximal Weighted Matching in General Graphs. **SIAM Journal of Computing**, v. 15, n. 1, p. 120-130, Feb. 1986.

GHIANI, G.; IMPROTA, G. Algorithm for the Hierarchical Chinese Postman Problem. **Operations Research Letters**, v. 26, n.1, p. 27-32, Feb. 2000.

GHIANI, G.; IMPROTA, G. An Efficient Transformation of the Generalized Vehicle Routing Problem. **European Journal of Operational Research**, v. 122, n. 1, p. 11-17, Apr. 2000.

GOLDBARG, M. C.; LUNA, H. P. L. **Otimização Combinatória e Programação Linear**. Rio de Janeiro: Editora Campus, 2000. 649p.

GOLDEN, B. L.; DE ARMON, J. S.; BAKER, E. K. Computational Experiments with Algorithms for a Class of Routing Problems. **Computers and Operations Research**, v. 10, n. 1, p. 47-59, 1983.

GOLDEN, B. L.; WONG, R. T. Capacitated Arc Routing Problems. **Networks**, v. 11, p. 305-315, 1981.

GONDRAN, M.; MINOUX, M. **Graphes et Algorithmes**. Paris: Eyrolles, 1985. 545p.

GREISTORFER, P. A. Tabu Scatter Search Metaheuristic for the Arc Routing Problem. **Computers and Industrial Engineering**, v. 44, n. 2, p. 249-266, Feb. 2002.

GUAN, M. On the Windy Postman Problem. **Discrete Applied Mathematics**, v. 9, n. 1, p. 41-46, Sep. 1984.

KEENAN, P. Spatial Decision Support Systems for the Vehicle Routing. **Decision Support Systems**, v. 22, n. 1, p. 65-71, Jan. 1998.

LAGUNA, M. Global Optimization and Meta-Heuristics. College of Business, University of Colorado at Boulder. Disponível em: < http://leeds.colorado.edu/faculty/laguna/articles/elss.pdf >. Acesso em: 18/08/2003.

LAPORTE, G. Modeling and Solving Several Classes of Arc Routing Problems as Travelling Salesman Problem. **Computers and Operations Research**, v. 24, n. 11, p. 1057-1061, Nov. 1997.

LETCHFORD, A. N.; EGLESE, R.W. The Rural Postman Problem with Deadline Classes. **European Journal of Operational Research**, v. 105, n. 3, p. 390-400, Mar. 1998.

LEVY, L.; BODIN, L. The Arc Oriented Location Routing Problem. **Infor**, v. 27, n. 1, p. 75-94, 1989.

LI, L.; EGLESE, R. An Interactive Algorithm for Vehicle Routeing for Winter-Gritting. **Journal of the Operational Research Society**, v. 47, n. 2, p. 217-228, 1996.

LIN, S.; KERNIGHAN, B. W. An Effective Heuristic Algorithm for the Traveling Salesman Problem. **Operations Research**, v. 21, p. 498-516, Mar-April 1973.

MARTI, R. Procedimientos Metaheurísticos en Optimización Combinatoria. A ser publicado em: **Publicacions Matemàtiques de la Universitat de Valencia**. Disponível em: < http://matheron.uv.es/investigar/metaheur.pdf. Acesso em: 18/08/2003.

MOURÃO, M. C. **Optimização de Rotas na Recolha de Resíduos Urbanos**. Lisboa, 1997. 215p. Tese de Doutoramento - Instituto Superior de Economia e Gestão, Universidade Técnica de Lisboa.

MUYLDERMANS, L.; CATTRYSSE, D.; Van OUDHEUSDEN, D.; LOTAN, T. Districting for Salt Spreading Operations. **European Journal of Operational Research**, v. 139, n. 3, p. 521-532, June 2002.

NEWELL, G. F.; DAGANZO, C. F. Design of Multiple-Vehicle Delivery Tours – I: A Ring-Radial Network. **Transportation Research**, v. 20, n. 5, p. 345-363, Oct. 1986.

NEWELL, G. F.; DAGANZO, C. F. Design of Multiple-Vehicle Delivery Tours – II: Other Metrics. **Transportation Research,** v. 20, n. 5, p. 365-376, Oct. 1986.

NEWELL, G. F.; DAGANZO, C. F. Design of Multiple-Vehicle Delivery Tours – III: Valuable Goods. **Transportation Research,** v. 20, n. 5, p. 377-390, Oct. 1986.

ORLOFF, C. A Fundamental Problem in Vehicle Routing. **Networks**, v. 4, p. 35-64, 1974.

PAPADIMITRIOU, C. H. On the Complexity of Edge Traversing. **Journal of the Association for Computing Machinery**, v. 23, n. 3, p. 544-554, July 1976.

PARTYKA, J. G.; HALL, R. W. On the Road to Service. **OR/MS Today**, Marietta GA Aug. 2000. Disponível em <http://www.lionhrtpub.com/orms/orms-8-00/vehiclerouting.html>. Acesso em: 8 jan. 2003.

PEARN, W. L. Approximate Solutions for the Capacitated Arc Routing Problem. **Computers and Operations Research**, v. 16, n. 6, p. 589-600, 1989.

PEARN, W. L. Augment-Insert Algorithms for the Capacitated Arc Routing Problem. **Computers and Operations Research**, v. 18, n. 2, p. 189-198, 1991.

PEARN, W. L.; CHOU, J. B. Improved Solutions for the Chinese Postman Problem on Mixed Networks. **Computers and Operations Research**, v. 26, n. 8, p. 819-827, July 1999.

PEARN, W. L.; LI, M. L. Algorithms for the Windy Postman Problem. **Computers and Operations Research**, v. 21, n. 6, p. 641-651, July 1994.

PEARN, W. L.; LIU, C. M. Algorithms for the Chinese Postman Problem on Mixed Networks. **Computers and Operations Research**, v. 22, n. 5, p. 479-489, May 1995.

PEARN, W. L.; WU, T. C. Algorithms for the Rural Postman Problem. **Computers and Operations Research**, v. 22, n. 8, p. 819-828, Oct. 1995.

PRINS, C. **Algorithmes de Graphes**. Paris: Eyrolles, 1994. 370p.

STERN, H.; DROR, M. Routing Electric Meter Readers. **Computers and Operations Research**, v. 6, n. 4, p. 209-223, 1979.

SYSLO, M. M.; DEO, N.; KOWALIK, J. **Discrete Optimization Algorithms**. New Jersey: Prentice-Hall, 1993, 542p.

TARANTILIS, C. D.; KIRANOUDIS, C.T. Using a Spatial Decision Support System for Solving the Vehicle Routing Problem. **Information and Management**, v. 39, n. 5, p. 359-375, Mar. 2002.

ULUSOY, G. The Fleet Size and Mix Problem for Capacitated Arc Routing. **European Journal of Operational Research**, v. 22, n. 3, p. 329-337, Dec. 1985.

WAN, H. F.; WEN, Y. P. Time-Constrained Chinese Postman Problems. **Computers and Mathematics with Applications**, v. 44, n. 3-4, p. 375-387, Aug. 2002.

WUNDERLICH, J.; COLLETTE, M.; LEVY, L.; BODIN, L. Scheduling Meter Readers for Southern California Gas Company. **Interfaces**, v. 22, n. 3, p. 22-30, May-June 1992.

APÊNDICE A – Estrutura de Dados Empregada na Implementação Computacional do Modelo

A estrutura de dados empregada na implementação computacional do modelo, bem como a inter-relação entre os seus elementos é mostrada na figura A.1 através de uma representação na forma de fluxograma.

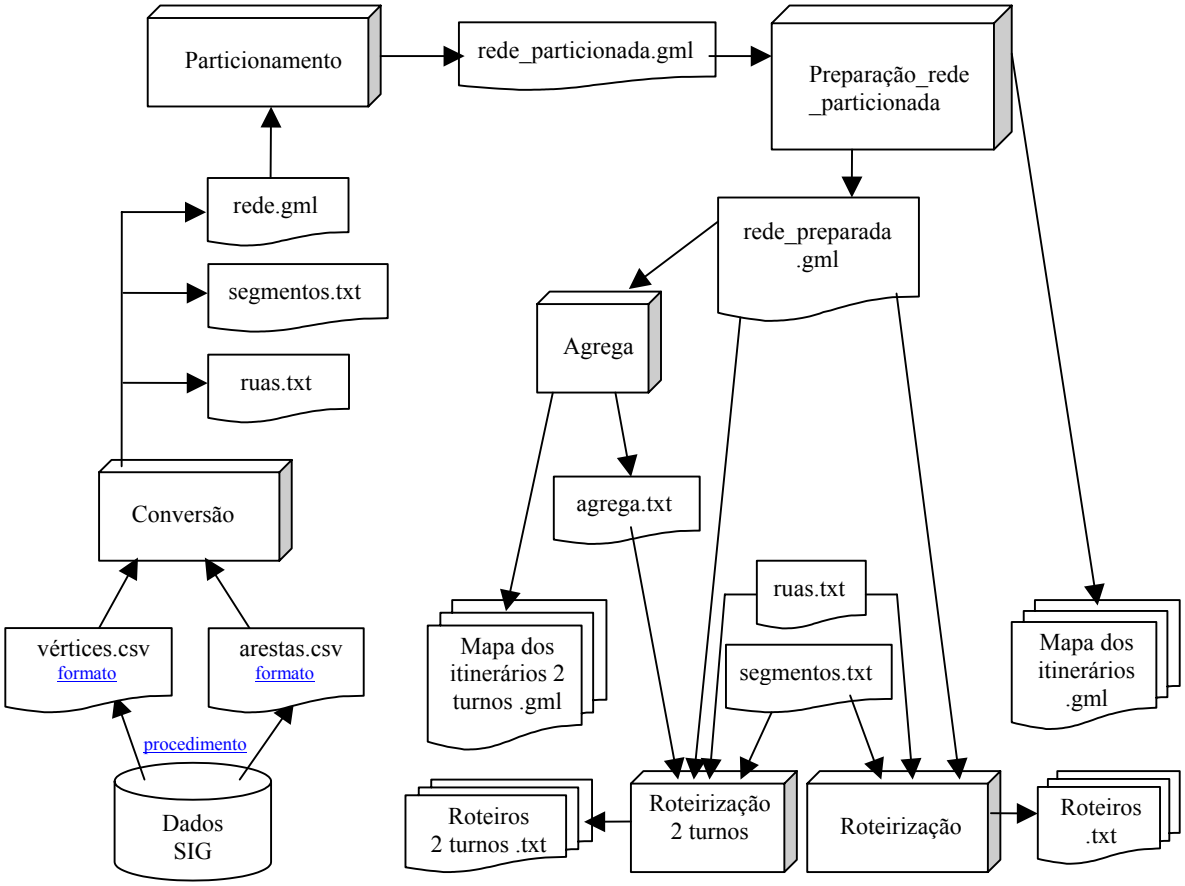


Figura A.1 – Estrutura de dados empregada na resolução do modelo

A descrição dos elementos que compõem a estrutura de dados é feita abaixo.

a) Dados SIG: contém a base de dados georeferenciada da rede de transportes, descrita pelas entidades vértice e aresta e os respectivos atributos;

b) Vértices.csv: arquivo texto gerado a partir do aplicativo SIG no formato csv (*comma separated value*), com as informações referentes aos vértices;

c) Arestas.csv: arquivo texto gerado a partir do aplicativo SIG no formato csv (*comma separated value*), com as informações referentes as arestas;

d) Conversão: realiza os procedimentos para gerar a partir dos arquivos vértices.csv e arestas.csv outros 3 arquivos (rede, segmentos e ruas).

e) Ruas.txt: arquivo texto no formato txt contendo o nome do logradouro de cada aresta.

f) Segmentos.txt: arquivo texto no formato txt contendo o número de aresta; a forma de atendimento (1 passada em zigue-zague ou uma face de quadra de cada vez); o tempo de percurso efetivo (com trabalho) e o tempo de percurso ocioso (sem trabalho).

g) Rede.gml: arquivo em formato texto que contem o registro dos vértices e das arestas, que será utilizado na execução do algoritmo de particionamento implementado em C++ e pelo visualizador de grafos (Graphlet);

h) Particionamento: procedimentos matemáticos implementados em C++ a partir dos quais são geradas as partições;

i) Rede_particionada.gml: arquivo em formato texto que contem o registro das partições geradas e que será utilizado no procedimento de preparação da rede implementado em C++ e pelo visualizador de grafos para representação gráfica da rede;

j) Preparação_rede_particionada: procedimentos matemáticos implementados em C++ que preparam cada uma das partições para resolução do PCC ou do PCR;

k) Rede_preparada.gml: arquivo em formato texto que contem o registro das partições já preparadas, que será utilizado na execução do procedimento de agregação de partições de um turno e/ou na resolução final (roteirização) do PCC ou do PCR. Este arquivo é utilizado também pelo visualizador de grafos;

l) Agrega: procedimento matemático implementado em C++ para a agregação de duas partições adjacentes, unidas em um vértice-depósito;

m) Agrega.txt: arquivo texto onde estão relacionados os pares de partições de um turno e os vértices-depósito que as unem;

n) Mapa dos itinerários 2 turnos: arquivo texto utilizado pelo visualizador de grafos para representar graficamente os pares de partições unidas pelos vértices-depósito determinadas anteriormente;

o) Roteirização 2 turnos: procedimento matemático implementado em C++ para determinação da seqüência de percurso em cada uma das partições de um turno;

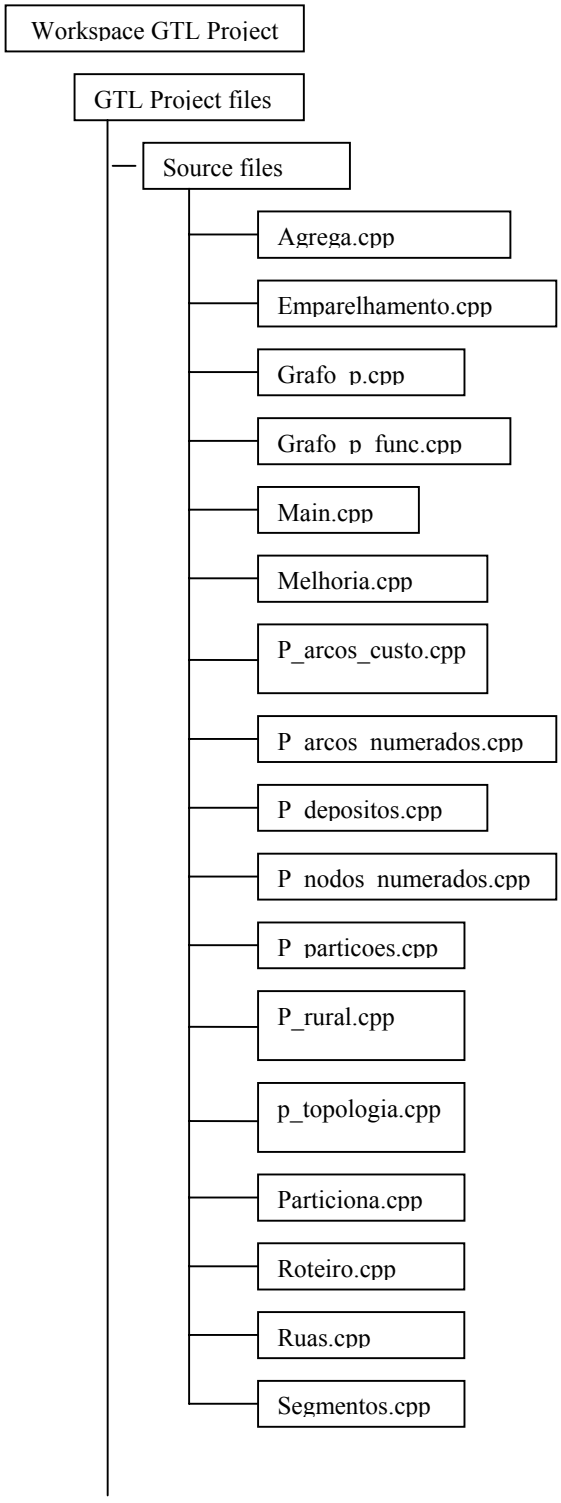
p) Roteiros 2 turnos.txt: arquivo texto contendo o relatório da seqüência a ser percorrida em cada um dos pares de partições;

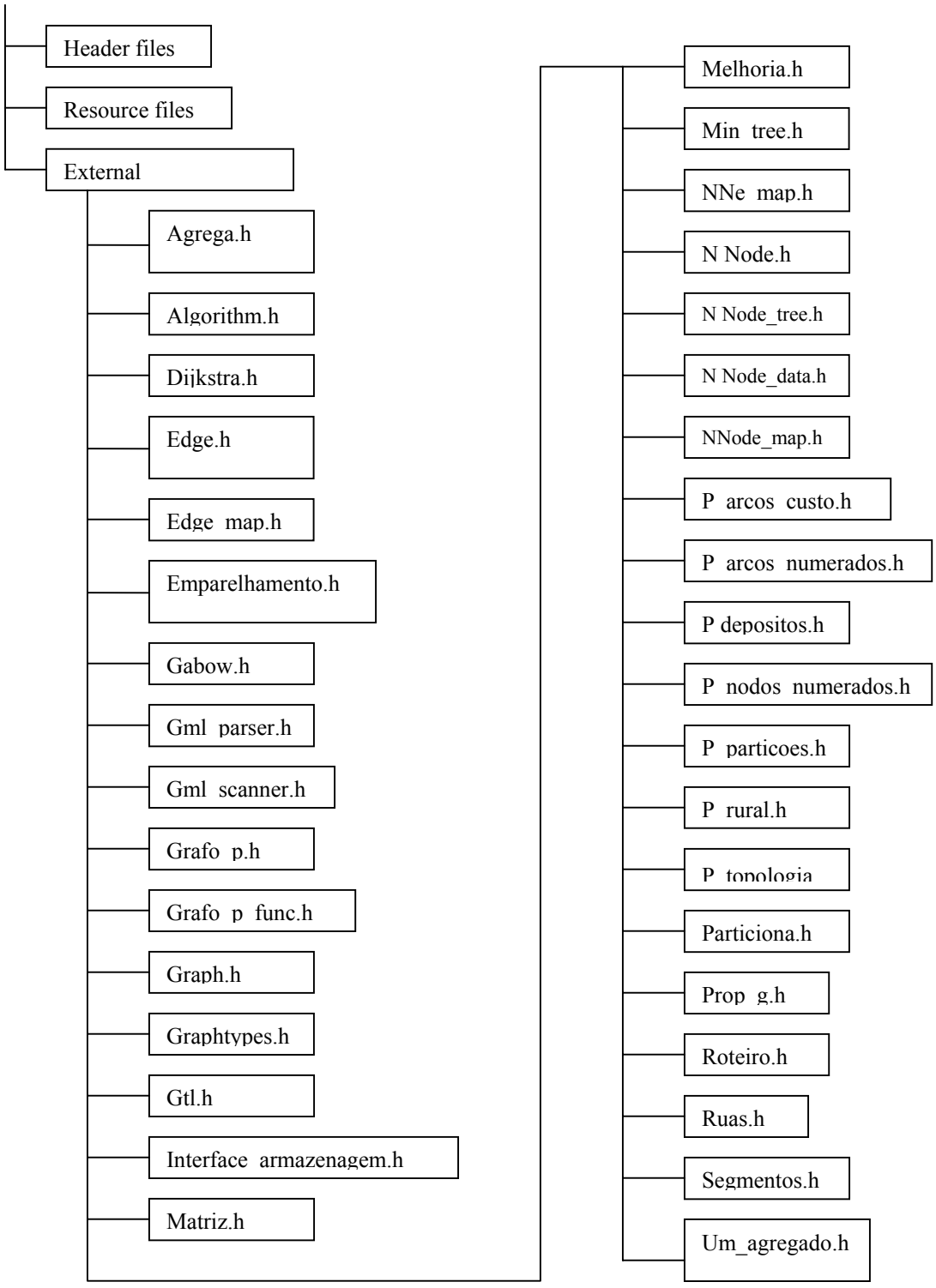
q) Mapa dos itinerários.gml: arquivo texto utilizado pelo visualizador de grafos para representar graficamente cada uma das partições com jornada de trabalho de um dia;

r) Roteirização: procedimento matemático implementado em C++ para a determinação da seqüência de percurso em cada uma das partições com jornada de trabalho de um dia;

s) Roteiros.txt: arquivo texto contendo o relatório da seqüência a ser percorrida em cada uma das partições com jornada de um dia de trabalho.

APÊNDICE B – Estrutura de Arquivos do Programa Implementado em C++




Header files
Resource files
External
Agrega.h
Algorithm.h
Dijkstra.h
Edge.h
Edge_map.h
Emparelhamento.h
Gabow.h
Gml_parser.h
Gml_scanner.h
Grafo_p.h
Grafo_p_func.h
Graph.h
Graphtypes.h
Gtl.h
Interface_armazenagem.h
Matriz.h
Melhoria.h
Min_tree.h
NNe_map.h
N Node.h
N Node_tree.h
N Node_data.h
NNode_map.h
P_arcos_custo.h
P_arcos_numerados.h
P depositos.h
P_nodos_numerados.h
P_particoes.h
P_rural.h
P_topologia
Particiona.h
Prop_g.h
Roteiro.h
Ruas.h
Segmentos.h
Um_agregado.h

APÊNDICE C – Programa Principal

```
#include <iostream>
#include <fstream>

#include <string>

#include <ctime>

// estrutura básica de grafos
#include "..\grafos3\grafo_p.h"
// funções extras para grafos
#include "..\grafos3\grafo_p_func.h"

// estruturas extras
#include "..\mapa3\segmento.h"
#include "..\mapa3\ruas.h"

// função de particionamento básica
#include "..\particiona3\particiona.h"
// função de melhoria do particionamento
#include "..\melhoria3\melhoria.h"

// função de emparelhamento
#include "..\emparelhamento\emparelhamento.h"

#include "gabow.h"

// função de agregação de partições
#include "..\agrega\agrega.h"

// função de roteirização (algoritmo de Fleury)
#include "..\roteiro\roteiro.h"

using namespace std;

// faz um arquivo com o relatório de partições do grafo
void relatorio_de_particoes(Grafo_p& grafo, string nome_arq) {
  cout << "Gerando relatorio de particoes " << nome_arq << flush;
  ofstream relatorio(nome_arq.c_str());
  gera_relatorio_particoes(grafo, relatorio);
  relatorio.close();
  cout << endl;
}

// tranforma um número inteiro em um string com o número em dois dígitos
string itos(int numero) {
  char numero_cstr[3];
  itoa(numero, numero_cstr, 10);
  if (numero_cstr[1] == '\0') {
    numero_cstr[2] = '\0';
```

```
    numero_cstr[1] = numero_cstr[0];
    numero_cstr[0] = '0';
  }
  return string(numero_cstr);
}


int main(int argc, char* argv[]) {

  //ofstream saida("saida.txt");
  //saida.sync_with_stdio();

  cout.rdbuf();

  string nome_arq;
  string nome_arq_original;

  // faz a leitura do nome do arquivo
  if (argc > 1) {
    // nome foi passado na linha de comando
    nome_arq_original = string(argv[1]);
  }
  else {
    cout << "Entre com a rede (.gml): ";
    cin >> nome_arq;
  }

  // coloca a extensão .gml ao fim do nome do arquivo,
  nome_arq_original = nome_arq + ".gml";

  // declara grfo com todas as propriedades
  Grafo_p grafo(
    P_TOPOLOGIA |
    P_RURAL |
    P_PARTICOES |
    P_DEPOSITOS |
    P_ARCOS_CUSTO |
    P_NODOS_NUMERADOS |
    P_ARCOS_NUMERADOS
  );

  // le o grafo
  grafo.le(nome_arq_original);

  double custo_total = grafo.p_arcos_custo->custo_total();

  // imprime informações sobre a rede lida
  cout << nome_arq_original << " lido, ok." << endl;
  cout << "Numero de vertices: " << grafo.number_of_nodes() << endl;
  cout << "Numero de arestas: " << grafo.number_of_edges() << endl;
  cout << "Tempo total de atendimento da rede: " << custo_total << "min." << endl;
```

```
  cout << "Tipo de configuracao dos roteiros" << endl;
  cout << "[1] Jornada de um dia" << endl;
  cout << "[2] Jornada de um turno" << endl;
  int jornada = 0;
  do {
    cout << "Informe o tipo de jornada: ";
    cin >> jornada;
  } while (jornada != 1 && jornada != 2);

  double custo_particao = 0.0;
  double min_custo_particao = min_max_custo_particao(grafo);

  int numero_de_particoes;
  do {

    do {
      cout << "Informe a jornada de trabalho desejada (minimo " << min_custo_particao << "), em minutos:
";
      cin >> custo_particao;
    } while (custo_particao < min_custo_particao);

    double numero_quebrado_particoes = custo_total / custo_particao;
    int parte_inteira = int(numero_quebrado_particoes);
    double parte_fracionaria = numero_quebrado_particoes - parte_inteira;
    numero_de_particoes = parte_inteira;

    int op; // opção digitada pelo usuário

    cout << "Numero de jornadas: " << numero_quebrado_particoes << endl;

    if (jornada == 1) {
      // jornada de um dia
      if (parte_fracionaria > 0.02) {
        cout << "Existiria uma jornada incompleta de " << custo_total - (parte_inteira * custo_particao) <<
"min." << endl;
        cout << "[1] Diluir nas demais (" << parte_inteira << " jornadas de " << custo_total / (parte_inteira)
<< "min)" << endl;
        cout << "[2] Criar uma nova jornada (" << parte_inteira+1 << " jornadas de " << custo_total /
(parte_inteira+1) << "min)" << endl;
        do {
          cout << "Escolha o que deseja fazer: ";
          cin >> op;
        } while (op < 1 || op > 2);
        switch (op) {
          case 1: numero_de_particoes = parte_inteira; break; // arredondamento para baixo
          case 2: numero_de_particoes = parte_inteira+1; break; // arredondamento para cima
        }
      }
    }
    else {
      // jornada de um turno
```

```
    if ((parte_inteira % 2) == 1) {
      cout << "Existiria numero impar de jornadas de um turno" << endl;
      cout << "[1] Diluir nas demais (" << parte_inteira-1 << " jornadas de " << custo_total / (parte_inteira-1) << "min)" << endl;
      cout << "[2] Criar uma nova jornada de um turno (" << parte_inteira+1 << " jornadas de " << custo_total / (parte_inteira+1) << "min)" << endl;
      do {
        cout << "Escolha o que deseja fazer: ";
        cin >> op;
      } while (op < 1 || op > 2);
      switch (op) {
        case 1: numero_de_particoes = parte_inteira-1; break;
        case 2: numero_de_particoes = parte_inteira+1; break;
      }
    }
    else {
      if (parte_fracionaria > 0.02) {
        cout << "Existiria uma jornada incompleta de " << custo_total - (parte_inteira * custo_particao) << "min." << endl;
        cout << "[1] Diluir nas demais (" << parte_inteira << " jornadas de " << custo_total / (parte_inteira) << "min)" << endl;
        cout << "[2] Criar duas novas jornadas de um turno (" << parte_inteira+2 << " jornadas de " << custo_total / (parte_inteira+2) << "min)" << endl;
        do {
          cout << "Escolha o que deseja fazer: ";
          cin >> op;
        } while (op < 1 || op > 2);
        switch (op) {
          case 1: numero_de_particoes = parte_inteira; break;
          case 2: numero_de_particoes = parte_inteira+2; break;
        }
      }
    }
  }

  custo_particao = custo_total / numero_de_particoes;

  // depois disso o custo da particao pode ter ficado menor do que devia
  if (custo_particao < min_custo_particao) {
    cout << "Custo da particao ficaria pequeno demais!" << endl;
  }
 } while (custo_particao < min_custo_particao);

 char rotular_nodos;
 cout << "Rotular nodos do mapa geral (s/n)? ";
 cin >> rotular_nodos;
 cout << endl;

 cout << "Serao geradas " << numero_de_particoes << " jornadas ";
 if (jornada == 1) cout << "de um dia"; else cout << "de um turno";
 cout << " (de " << custo_particao << "min)" << endl;
```

```
cout << "Tecle <Enter> para iniciar com esses valores. <Ctrl C> para cancelar.";
cin.ignore(); cin.ignore(); // um para tirar o enter anterior do buffer e outro para o próximo
cout << endl;

// inicia contagem de tempo de execucao do algoritmo
int t0 = time(0);

cout << "Gerando particoes iniciais..." << flush;
particiona_tudo(grafo, custo_particao);
cout << endl;

cout << "Agregando particoes menores..." << flush;
agrega_particoes_pequenas(grafo, numero_de_particoes);
cout << endl;

// Gera relatório de partições antes da melhoria
relatorio_de_particoes(grafo, nome_arq + "_relatorio_particoes_antes_da_melhoria.txt");

cout << "Executando melhoria..." << endl;
melhoria(grafo); cout << endl;

relatorio_de_particoes(grafo, nome_arq + "_relatorio_particoes.txt");

// colore as partições do grafo
grafo.p_particoes->colore();
// isso faz arestas duplas e triplas aparecerem na visualização
// (se não elas ficariam sobrepostas)
grafo.p_topologia->calcula_ancoras();

// salva grafo particionado
string nome_arq_particionado = nome_arq + "_particionado.gml";
cout << "Salvando grafo particionado " << nome_arq_particionado << flush;
grafo.p_nodos_numerados->rotula(rotular_nodos == 's' || rotular_nodos == 'S');
grafo.save(nome_arq_particionado.c_str());
cout << endl;

// esse arquivo será usado quando for criado os relatórios de percurso
cout << "Lendo arquivo de ruas..." << flush;
Ruas ruas;
ruas.le("ruas.txt");
cout << endl;

// certifica que o número de partições geradas é igual ao número de partições desejado
assert (grafo.p_particoes->num_particoes() == numero_de_particoes);

// vetor com todos os grafos das partições
vector<Grafo_p*> particao;

// grafo de toda a rede com as partições já preparadas
Grafo_p grafo_particionado_preparado(grafo.get_propriedades() | P_PARTICOES);
```

```
  // para cada partição, transforma o grafo em grafo euleriano
  int i;
  double custo_adicional_total = 0.0;
  for (i = 1; i <= numero_de_particoes; ++i) {

    particao.push_back(new Grafo_p(grafo.get_propriedades() | P_PARTICOES));
    grafo.p_particoes->extrai(*particao[i-1], i);

    string numero = itos(i); // string com o número da partição
    particao[i-1]->p_nodos_numerados->rotula(true); // rotula vértices das partições
    particao[i-1]->p_arcos_custo->rotula(false); // não rotula arcos com o custo

    // salva partição antes da preparação
    particao[i-1]->p_topologia->calcula_ancoras(); // mostra arestas duplas e triplas
    string nome_arq_particao = "partições\\" + nome_arq + "_particao" + numero + ".gml";
    cout << "Salvando " << nome_arq_particao << flush;
    particao[i-1]->save(nome_arq_particao.c_str());
    cout << endl;

    cout << "Preparando particao" << i << "..." << flush;
    double custo_adicional = eulerizacao(*particao[i-1]);
    cout << endl;
    cout << "custo adicional (percurso ocioso) = " << custo_adicional << endl;
    custo_adicional_total += custo_adicional;

    uniao(grafo_particionado_preparado, *particao[i-1]);

  }


  // salva grafo particionado preparado
  grafo_particionado_preparado.p_topologia->calcula_ancoras();
  string nome_arq_particionado_preparado = nome_arq + "_particionado_preparado.gml";
  cout << "Salvando grafo particionado preparado " << nome_arq_particionado_preparado << flush;
  grafo_particionado_preparado.p_nodos_numerados->rotula(rotular_nodos == 's' || rotular_nodos == 'S');
  grafo_particionado_preparado.save(nome_arq_particionado_preparado.c_str());
  cout << endl;

  // Gera relatório de partições depois da preparação
  relatorio_de_particoes(grafo_particionado_preparado,                    nome_arq                    +
"_relatorio_particoes_preparadas.txt");

  if (jornada == 1) {
    // faz os roteiros

    for (i = 1; i <= numero_de_particoes; ++i) {

      string numero = itos(i); // string com o número da partição
      // salva mapa da partição preparada
      particao[i-1]->p_topologia->calcula_ancoras();
```

```
    string nome_arq_particao_preparada = "partições\\" + nome_arq + "_particao" + numero +
"_preparada.gml";
    cout << "Salvando " << nome_arq_particao_preparada << flush;
    particao[i-1]->save(nome_arq_particao_preparada.c_str());
    cout << endl;

    cout << "Lendo arquivo de segmentos (segmentos.txt)..." << flush;
    Segmentos segmentos(*particao[i-1]);
    segmentos.Interface_armazenagem::le("segmentos.txt");
    cout << endl;

    Roteiro rot;
    rot.reset();
    rot.set_segmentos(&segmentos);

    string nome_arq_percurso = "partições\\" + nome_arq + "_roteiro" + numero + ".txt";
    ofstream arq_percurso(nome_arq_percurso.c_str());
    cout << "Salvando roteiro em " << nome_arq_percurso << flush;

    // preferencialemtne começa por um vértice depósito
    if (particao[i-1]->tem_propriedade(P_DEPOSITOS)) {
      graph::node_iterator ni;
      for (ni = particao[i-1]->nodes_begin(); ni != particao[i-1]->nodes_end(); ++ni) {
        if (particao[i-1]->p_depositos->get(*ni)) {
          rot.set_inicio(*ni);
          break;
        }
      }
    }

    if (rot.check(*particao[i-1]) != algorithm::GTL_OK) {
      cout << " *** ERRO: Nao foi possivel criar um roteiro para a particao " << i << " ***" << endl;
      assert(false);
      exit(1);
    }
    rot.run(*particao[i-1]); // aplica algoritmo de Fleury
    arq_percurso << "Roteiro particao " << i << endl;
    arq_percurso << "Tempo do percurso: " << particao[i-1]->p_arcos_custo->custo_total() << "min." <<
endl << endl;
    rot.relatorio(arq_percurso, *particao[i-1], ruas);
    arq_percurso.close();
    cout << endl;

   } // fim da roteirização de um itinerário de um dia

  } // fim da geração dos mapas e dos roteiros para itinerários de um dia

  else {
   // agrega e faz os roteiros (2 turnos)

   Agrega agregacoes;
```

```
 cout << "Calculando a composição dos turnos..." << endl;
// calcula os pares de partições que serão agregadas
agregacoes.agrega_particoes(grafo_particionado_preparado);


// cria os mapas das partições agregadas
set<int> part_ja_foi;
int n = agregacoes.numero_agregados() * 2;
for (i = 1; i <= numero_de_particoes; ++i) {
  if (part_ja_foi.find(i) != part_ja_foi.end()) continue;

  int p[2];
  p[0] = i;
  p[1] = agregacoes.get_agregado(p[0]);
  part_ja_foi.insert(p[0]);
  part_ja_foi.insert(p[1]);

  Grafo_p g2p(grafo_particionado_preparado.get_propriedades()); // grafo duas particoes
  agregacoes.get_particao_agregada(g2p, grafo_particionado_preparado, p[0]);

  string num_p1 = itos(p[0]);
  string num_p2 = itos(p[1]);

  string nome_arq_particao = "partições\\" + nome_arq + "_particao" + num_p1 + "-" + num_p2 +
"_preparada.gml";
  g2p.p_nodos_numerados->rotula(true);
  g2p.save(nome_arq_particao.c_str());

  // roteiriza
  string nome_arq_percurso = "partições\\" + nome_arq + "_roteiro" + num_p1 + "-" + num_p2 + ".txt";
  ofstream arq_percurso(nome_arq_percurso.c_str());
  cout << "Salvando roteiro em " << nome_arq_percurso << endl;
  arq_percurso << "Roteiro partição agregada " << p[0] << "-" << p[1] << endl;
  arq_percurso << "Tempo do percurso: ";
  arq_percurso << particao[p[0]-1]->p_arcos_custo->custo_total() << " + " << particao[p[1]-1]-
>p_arcos_custo->custo_total() << " = ";
  arq_percurso << particao[p[0]-1]->p_arcos_custo->custo_total() + particao[p[1]-1]->p_arcos_custo-
>custo_total() << "min." << endl << endl;

  int np;
  // cria roteiro para cada turno
  for (np = 0; np < 2; ++np) {
    Roteiro rot;
    rot.reset();

    if (np == 0) {
      arq_percurso << "Primeiro turno" << endl;
    }
    else {
      arq_percurso << "Segundo turno" << endl;
    }
```

```
      cout << "Lendo arquivo de segmentos (segmentos.txt)..." << flush;
      Segmentos segmentos(*particao[p[np]-1]);
      segmentos.Interface_armazenagem::le("segmentos.txt");
      cout << endl;
      rot.set_segmentos(&segmentos);

      if (particao[p[np]-1]->tem_propriedade(P_NODOS_NUMERADOS)) {
        rot.set_inicio(particao[p[np]-1]->p_nodos_numerados->get(agregacoes.get_deposito(p[np])));
      }

      if (rot.check(*particao[p[np]-1]) != algorithm::GTL_OK) {
        cout << " *** ERRO: Nao foi possivel criar um roteiro para a particao " << i << " ***" << endl;
        assert(false);
        exit(1);
      }

      rot.run(*particao[p[np]-1]); // aplica algoritmo de Fleury

      rot.relatorio(arq_percurso, *particao[p[np]-1], ruas);
      arq_percurso << endl;

    }

    arq_percurso.close();

  } // fim da roteirização de um itinerário de dois turnos

} // fim agrega e roteiriza dois turnos

// desaloca os grafos das partições
for (i=0; i<particao.size(); ++i) {
  delete particao[i];
}

cout << "Custo adicional total " << custo_adicional_total << endl;

// mostra relatório final das partições na tela
gera_relatorio_particoes(grafo_particionado_preparado, cout);

int tf = time(0);
cout << "Tempo de execucao: " << tf-t0 << "s." << endl;

return 0;

}
```

APÊNDICE D – Listagem da Execução do Modelo em sua Aplicação no Estudo de Caso

Execução do primeiro cenário

```
Informe a rede de atendimento (.gml): logradouros.gml
logradouros.gml lido, ok.
Numero de nodos: 293
Numero de segmentos: 622
Tempo total de atendimento da posicao: 6646min.

Informe o tipo de configuração dos roteiros
[1] Jornada de um dia
[2] Jornada de um turno
Informe a opcao desejada:1

PARTICIONAMENTO
Informe a jornada de trabalho desejada (minimo 113.4), em minutos: Numero de jornadas: 13.8458
Existe uma particao incompleta.
[1] Diluir nas demais particoes (13 jornadas de 511.231min)
[2] Criar uma nova particao (14 jornadas de 474.714min)
Escolha o que deseja fazer: 2

Jornada de trabalho: 474.714min.
Numero de jornadas: 14
Rotular nodos do mapa (s/n)? n

Tecle <Enter> para iniciar com esses valores. <Ctrl C> para cancelar.

Gerando particoes iniciais...
Agregando particoes menores...
Gerando relatorio de particoes antes da melhoria logradouros_relatorio_particoes_antes_da_melhoria.txt
Executando melhoria...
(87-117)  de 13 para 9  ganho: 61
(118-117)  de 13 para 11  ganho: 50
(218-244)  de 6 para 8  ganho: 31
(223-236)  de 10 para 14  ganho: 27.2
(207-223)  de 10 para 14  ganho: 38
(185-203)  de 11 para 10  ganho: 50.2
(141-140)  de 9 para 11  ganho: 40.8
(119-141)  de 13 para 11  ganho: 39.6
(94-95)  de 12 para 14  ganho: 25.6
(161-160)  de 9 para 7  ganho: 20.2
(189-207)  de 12 para 14  ganho: 16.4
(167-189)  de 12 para 14  ganho: 21.4
(145-167)  de 12 para 14  ganho: 26
(123-145)  de 12 para 14  ganho: 21.2
(119-141)  de 11 para 12  ganho: 33.6
(135-157)  de 1 para 5  ganho: 15
(200-216)  de 6 para 7  ganho: 14.6
(119-118)  de 13 para 11  ganho: 13.6
(233-245)  de 6 para 10  ganho: 11.2
(88-118)  de 11 para 13  ganho: 11
(70-92)  de 13 para 12  ganho: 9
(203-202)  de 8 para 10  ganho: 6.4
```

(93-94) de 12 para 14 ganho: 5.2
(93-123) de 12 para 14 ganho: 45.6
(120-119) de 13 para 12 ganho: 13.4
(53-52) de 1 para 13 ganho: 9.4
(222-223) de 10 para 14 ganho: 5.2
(222-235) de 10 para 14 ganho: 70.8
(206-222) de 10 para 14 ganho: 23.4
(186-185) de 11 para 10 ganho: 52
(222-221) de 10 para 14 ganho: 37.2
(139-161) de 9 para 11 ganho: 35
(161-160) de 7 para 11 ganho: 20.2
(53-52) de 13 para 9 ganho: 19.6
(188-206) de 10 para 14 ganho: 17.6
(164-186) de 11 para 10 ganho: 20.2
(161-183) de 8 para 11 ganho: 31.2
(206-205) de 10 para 14 ganho: 27.4
(200-216) de 7 para 8 ganho: 14.6
(178-192) de 3 para 7 ganho: 12.6
(181-199) de 6 para 8 ganho: 11
(199-198) de 4 para 8 ganho: 9.8
(122-123) de 12 para 14 ganho: 5
(89-119) de 13 para 11 ganho: 4.6
(219-227) de 8 para 10 ganho: 2.4
(219-218) de 8 para 10 ganho: 2.8
(199-215) de 6 para 8 ganho: 19.8
(214-213) de 4 para 6 ganho: 7.8
(135-157) de 5 para 4 ganho: 9
(205-221) de 10 para 14 ganho: 5.2
(153-172) de 3 para 5 ganho: 4.4

Gerando relatorio de particoes logradouros_relatorio_particoes.txt
Salvando grafo particionado logradouros_particionado.gml

EMPARELHAMENTO
Lendo arquivo de ruas...

Extraindo particao 1ok! Salvando partições\logradouros_particao01.gml
Executando emparelhamento...(46-48) : custo = 0.014
(81-79) : custo = 0.014
(49-82) : custo = 0.018
(102-113) : custo = 0.004
(83-115) : custo = 0.03
(52-85) : custo = 0.02

Custo adicional: 10.1
Salvando partições\logradouros_particao01_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao01_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 2ok! Salvando partições\logradouros_particao02.gml
Executando emparelhamento...(98-74) : custo = 0.008
(72-58) : custo = 0.008
(99-106) : custo = 0.01
(54-45) : custo = 0.008
(23-24) : custo = 0.016

(37-39) : custo = 0.048
(61-76) : custo = 0.01

Custo adicional: 10.908
Salvando partições\logradouros_particao02_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao02_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 3ok! Salvando partições\logradouros_particao03.gml
Executando emparelhamento...(170-175) : custo = 0.006
(176-194) : custo = 0.028
(178-154) : custo = 0.044
(174-156) : custo = 0.006

Custo adicional: 8.484
Salvando partições\logradouros_particao03_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao03_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 4ok! Salvando partições\logradouros_particao04.gml
Executando emparelhamento...(230-239) : custo = 0.014
(196-212) : custo = 0.014
(232-248) : custo = 0.012

Custo adicional: 4.04
Salvando partições\logradouros_particao04_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao04_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 5ok! Salvando partições\logradouros_particao05.gml
Executando emparelhamento...(153-172) : custo = 0.006
(155-133) : custo = 0.014

Custo adicional: 2.02
Salvando partições\logradouros_particao05_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao05_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 6ok! Salvando partições\logradouros_particao06.gml
Executando emparelhamento...(213-240) : custo = 0.074
(241-226) : custo = 0.016
(256-257) : custo = 0.01
(216-217) : custo = 0.016

Custo adicional: 11.716
Salvando partições\logradouros_particao06_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao06_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...
Extraindo particao 7ok! Salvando partições\logradouros_particao07.gml
Executando emparelhamento...(178-192) : custo = 0.006
(181-160) : custo = 0.03

Custo adicional: 3.636
Salvando partições\logradouros_particao07_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao07_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 8ok! Salvando partições\logradouros_particao08.gml
Executando emparelhamento...(199-198) : custo = 0.016
(200-216) : custo = 0.014
(244-217) : custo = 0.038
(202-184) : custo = 0.016

Custo adicional: 8.484
Salvando partições\logradouros_particao08_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao08_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 9ok! Salvando partições\logradouros_particao09.gml
Executando emparelhamento...(52-116) : custo = 0.052
(85-86) : custo = 0.016

Custo adicional: 6.868
Salvando partições\logradouros_particao09_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao09_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 10ok! Salvando partições\logradouros_particao10.gml
Executando emparelhamento...(245-273) : custo = 0.106
(202-204) : custo = 0.032
(219-185) : custo = 0.028
(234-231) : custo = 0.006

Custo adicional: 17.372
Salvando partições\logradouros_particao10_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao10_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 11ok! Salvando partições\logradouros_particao11.gml
Executando emparelhamento...(184-185) : custo = 0.016
(118-89) : custo = 0.03

Custo adicional: 4.646
Salvando partições\logradouros_particao11_emparelhado.gml
PERCURSO
Salvando caminho em partições\logradouros_particao11_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...
Extraindo particao 12ok! Salvando partições\logradouros_particao12.gml
Executando emparelhamento...(122-70) : custo = 0.022
(121-91) : custo = 0.014
(120-119) : custo = 0.018

Custo adicional: 5.454
Salvando partições\logradouros_particao12_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao12_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 13ok! Salvando partições\logradouros_particao13.gml
Executando emparelhamento...(118-65) : custo = 0.024
(66-11) : custo = 0.038
(90-91) : custo = 0.018
(120-43) : custo = 0.04
(64-40) : custo = 0.014
(68-70) : custo = 0.016
(30-7) : custo = 0.024

Custo adicional: 17.574
Salvando partições\logradouros_particao13_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao13_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 14ok! Salvando partições\logradouros_particao14.gml
Executando emparelhamento...Extraindo componentes conexas...6
5

Numero de componentes: 5
Criando Gc...
  criando nodos...

(221-205) : custo = 0.014
(206-207) : custo = 0.044
(94-125) : custo = 0.032
(252-253) : custo = 0.018
(209-168) : custo = 0.046
Solucao 0: custo 32.754

(221-205) : custo = 0.014
(206-207) : custo = 0.044
(95-125) : custo = 0.014
(252-253) : custo = 0.018
(209-169) : custo = 0.028
Solucao 1: custo 29.118

(221-205) : custo = 0.014
(206-207) : custo = 0.044
(95-125) : custo = 0.014
(209-168) : custo = 0.046
Solucao 2: custo 29.118

(221-205) : custo = 0.014
(206-207) : custo = 0.044
(94-125) : custo = 0.032
(279-280) : custo = 0.016
(209-168) : custo = 0.046
Solucao 3: custo 26.152

(221-205) : custo = 0.014
(206-207) : custo = 0.044
(93-94) : custo = 0.016

(95-125) : custo = 0.014
(209-169) : custo = 0.028
Solucao 4: custo 29.116

(221-205) : custo = 0.014
(206-207) : custo = 0.044
(94-125) : custo = 0.032
(252-253) : custo = 0.018
(209-168) : custo = 0.046
Solucao 5: custo 32.754
Melhor solucao: 3

Custo adicional: 26.152
Salvando partições\logradouros_particao14_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao14_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Gerando relatorio de particoes emparelhado logradouros_relatorio_particoes.txt
Salvando grafo particionado emparelhado logradouros_particionado_emparelhado.gml
Custo adicional total 137.454

Part	Arcos	Custo	Vertices
1	74	493.1	36
2	88	490.9	50
3	32	485.7	16
4	36	481.2	19
5	40	478.4	17
6	57	488.7	34
7	26	484.8	13
8	28	478.3	13
9	36	480	15
10	58	472.5	28
11	34	476.2	16
12	42	481.1	17
13	90	493.5	42
14	70	472.5	44
Tot.	711	6756.9	360*
Med.	50.8	482.64	25.7
Desv.	22	7.026	13

* vértices que compartilham partições são contados uma vez por partição
Tempo de execucao: 10s.
FIM
?

Execução do segundo cenário

Informe a rede de atendimento (.gml): logradouros.gml
logradouros.gml lido, ok.
Numero de nodos: 293
Numero de segmentos: 622
Tempo total de atendimento da posicao: 6646min.

Informe o tipo de configuração dos roteiros
[1] Jornada de um dia

```
[2] Jornada de um turno
Informe a opcao desejada: 2

PARTICIONAMENTO
Informe a jornada de trabalho desejada (minimo 113.4), em minutos: 240
Numero de jornadas: 27.6917
Existe uma particao incompleta.
[1] Diluir nas demais particoes (26 jornadas de 255.615min)
[2] Criar uma nova particao (28 jornadas de 237.357min)
Escolha o que deseja fazer: 2

Jornada de trabalho: 237.357min.
Numero de jornadas: 28
Rotular nodos do mapa (s/n)? n

Tecle <Enter> para iniciar com esses valores. <Ctrl C> para cancelar.

Gerando particoes iniciais...
Agregando particoes menores...
Gerando relatorio de particoes antes da melhoria logradouros_relatorio_particoes_antes_da_melhoria.txt
Executando melhoria...
(120-142)  de 24 para 26  ganho: 66.2
(206-222)  de 17 para 25  ganho: 41.6
(88-87)  de 23 para 16  ganho: 40.2
(119-141)  de 22 para 24  ganho: 31.8
(141-140)  de 19 para 22  ganho: 40.8
(225-238)  de 27 para 28  ganho: 30.8
(38-37)  de 2 para 15  ganho: 30.2
(65-64)  de 23 para 16  ganho: 28.2
(88-87)  de 16 para 19  ganho: 40.2
(160-159)  de 11 para 14  ganho: 27.8
(157-179)  de 8 para 10  ganho: 27.4
(91-121)  de 23 para 26  ganho: 27.4
(66-89)  de 23 para 19  ganho: 27.2
(161-183)  de 19 para 22  ganho: 36.4
(94-95)  de 27 para 28  ganho: 25.6
(242-241)  de 7 para 11  ganho: 25.2
(218-217)  de 11 para 18  ganho: 57
(242-255)  de 7 para 11  ganho: 37.8
(180-198)  de 11 para 12  ganho: 31.6
(203-202)  de 18 para 20  ganho: 26.4
(143-142)  de 24 para 26  ganho: 23.2
(165-187)  de 25 para 24  ganho: 21
(144-143)  de 24 para 26  ganho: 20.4
(30-29)  de 23 para 16  ganho: 18.4
(110-109)  de 9 para 2  ganho: 16.6
(41-65)  de 23 para 16  ganho: 15.4
(110-132)  de 5 para 2  ganho: 15.2
(187-205)  de 17 para 25  ganho: 15
(166-165)  de 25 para 24  ganho: 17.2
(143-165)  de 24 para 26  ganho: 18
(141-163)  de 22 para 24  ganho: 26.8
(30-41)  de 23 para 16  ganho: 14.6
(87-117)  de 16 para 19  ganho: 24.4
(139-161)  de 19 para 14  ganho: 46
(42-41)  de 23 para 16  ganho: 16
```

(206-205) de 21 para 25 ganho: 13.8
(120-119) de 23 para 19 ganho: 13.4
(90-120) de 23 para 26 ganho: 18.8
(91-90) de 23 para 26 ganho: 21.4
(68-91) de 23 para 26 ganho: 14
(69-68) de 23 para 26 ganho: 17
(120-119) de 19 para 24 ganho: 13.4
(178-192) de 4 para 5 ganho: 12.8
(214-213) de 11 para 4 ganho: 12
(226-241) de 7 para 6 ganho: 12
(241-254) de 7 para 6 ganho: 25
(241-254) de 6 para 11 ganho: 15.6
(85-84) de 12 para 1 ganho: 11.6
(39-63) de 1 para 15 ganho: 13.4
(39-55) de 9 para 15 ganho: 11.4
(55-62) de 9 para 15 ganho: 17.8
(25-24) de 15 para 2 ganho: 12.8
(110-132) de 2 para 9 ganho: 34.2
(85-115) de 12 para 1 ganho: 9.6
(218-244) de 17 para 21 ganho: 9.4
(93-94) de 27 para 28 ganho: 5.2
(93-123) de 27 para 28 ganho: 25.6
(123-145) de 27 para 28 ganho: 21.2
(145-167) de 27 para 28 ganho: 26
(167-189) de 27 para 28 ganho: 6
(183-201) de 18 para 22 ganho: 5.2
(141-140) de 22 para 24 ganho: 8.4
(222-223) de 17 para 27 ganho: 5.2
(206-222) de 25 para 27 ganho: 43.4
(166-165) de 24 para 25 ganho: 25
(144-143) de 26 para 25 ganho: 19.8
(189-207) de 27 para 28 ganho: 16.4
(222-235) de 17 para 27 ganho: 8.2
(243-249) de 11 para 17 ganho: 57.6
(207-223) de 27 para 28 ganho: 33.2
(214-213) de 4 para 11 ganho: 20.6
(226-241) de 6 para 11 ganho: 12
(215-226) de 6 para 11 ganho: 14.6
(218-244) de 21 para 17 ganho: 9
(222-221) de 17 para 27 ganho: 26.8
(122-123) de 25 para 28 ganho: 5
(122-144) de 25 para 28 ganho: 26.4
(153-172) de 3 para 5 ganho: 4.4
(159-181) de 11 para 13 ganho: 4.4
(160-159) de 14 para 13 ganho: 7.8
(150-148) de 5 para 2 ganho: 4.2
(151-150) de 5 para 2 ganho: 9.2
(152-130) de 2 para 5 ganho: 24
(133-155) de 5 para 9 ganho: 5.2
(62-77) de 9 para 15 ganho: 8.2
(13-25) de 15 para 2 ganho: 10.8
(156-174) de 5 para 9 ganho: 8.4
(173-172) de 3 para 5 ganho: 16.6
(178-192) de 5 para 4 ganho: 16
(174-178) de 5 para 9 ganho: 9.6
(249-256) de 7 para 17 ganho: 4

(137-159) de 14 para 12 ganho: 2.8
(115-114) de 12 para 1 ganho: 38
(113-135) de 1 para 10 ganho: 13.2
(63-79) de 1 para 15 ganho: 3.6
(79-78) de 1 para 15 ganho: 4.8
(78-112) de 1 para 9 ganho: 12.8
(133-155) de 9 para 5 ganho: 6.8
(250-257) de 7 para 17 ganho: 2.2
(257-256) de 7 para 17 ganho: 13.8
(257-262) de 7 para 17 ganho: 4.2
(262-271) de 7 para 17 ganho: 4.2
(271-276) de 7 para 17 ganho: 16.2
(204-220) de 17 para 21 ganho: 4.2
(220-228) de 17 para 21 ganho: 8.8
(197-213) de 4 para 6 ganho: 2
(219-227) de 17 para 21 ganho: 1
(122-121) de 26 para 28 ganho: 0.2
(68-67) de 23 para 26 ganho: 14.2
(42-41) de 16 para 23 ganho: 16
(67-90) de 23 para 26 ganho: 4

Gerando relatorio de particoes logradouros_relatorio_particoes.txt
Salvando grafo particionado logradouros_particionado.gml

EMPARELHAMENTO
Lendo arquivo de ruas...

Extraindo particao 1ok! Salvando partições\logradouros_particao01.gml
Executando emparelhamento...(46-48) : custo = 0.014
(81-79) : custo = 0.014
(49-52) : custo = 0.022
(82-83) : custo = 0.004
(102-113) : custo = 0.004

Custo adicional: 5.858
Salvando partições\logradouros_particao01_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao01_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...
Extraindo particao 2ok! Salvando partições\logradouros_particao02.gml
Executando emparelhamento...(98-74) : custo = 0.008
(72-58) : custo = 0.008
(99-106) : custo = 0.01
(54-45) : custo = 0.008
(23-37) : custo = 0.024
(130-110) : custo = 0.044
(24-25) : custo = 0.008
(131-76) : custo = 0.04

Custo adicional: 15.15
Salvando partições\logradouros_particao02_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao02_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 3ok! Salvando partições\logradouros_particao03.gml

Executando emparelhamento...(170-175) : custo = 0.006
(176-172) : custo = 0.008
(194-173) : custo = 0.018

Custo adicional: 3.232
Salvando partições\logradouros_particao03_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao03_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 4ok! Salvando partições\logradouros_particao04.gml
Executando emparelhamento...(192-178) : custo = 0.006
(196-197) : custo = 0.02

Custo adicional: 2.626
Salvando partições\logradouros_particao04_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao04_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 5ok! Salvando partições\logradouros_particao05.gml
Executando emparelhamento...(130-131) : custo = 0.014
(173-174) : custo = 0.032
(153-154) : custo = 0.018
(155-133) : custo = 0.014

Custo adicional: 7.878
Salvando partições\logradouros_particao05_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao05_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 6ok! Salvando partições\logradouros_particao06.gml
Executando emparelhamento...(197-212) : custo = 0.032
(248-232) : custo = 0.012
(239-230) : custo = 0.014
(240-226) : custo = 0.036
Custo adicional: 9.494
Salvando partições\logradouros_particao06_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao06_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...
Extraindo particao 7ok! Salvando partições\logradouros_particao07.gml
Executando emparelhamento...(254-255) : custo = 0.016
(256-276) : custo = 0.054
Custo adicional: 7.07
Salvando partições\logradouros_particao07_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao07_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 8ok! Salvando partições\logradouros_particao08.gml
Executando emparelhamento...
Custo adicional: 0
Salvando partições\logradouros_particao08_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao08_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 9ok! Salvando partições\logradouros_particao09.gml
Executando emparelhamento...(110-76) : custo = 0.016
(178-156) : custo = 0.012

Custo adicional: 2.828
Salvando partições\logradouros_particao09_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao09_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 10ok! Salvando partições\logradouros_particao10.gml
Executando emparelhamento...
Custo adicional: 0
Salvando partições\logradouros_particao10_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao10_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 11ok! Salvando partições\logradouros_particao11.gml
Executando emparelhamento...(241-254) : custo = 0.014
(255-217) : custo = 0.052
(213-200) : custo = 0.062
(199-198) : custo = 0.016

Custo adicional: 14.544
Salvando partições\logradouros_particao11_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao11_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 12ok! Salvando partições\logradouros_particao12.gml
Executando emparelhamento...
Custo adicional: 0
Salvando partições\logradouros_particao12_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao12_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 13ok! Salvando partições\logradouros_particao13.gml
Executando emparelhamento...(181-160) : custo = 0.03

Custo adicional: 3.03
Salvando partições\logradouros_particao13_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao13_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 14ok! Salvando partições\logradouros_particao14.gml
Executando emparelhamento...(115-116) : custo = 0.018

Custo adicional: 1.818
Salvando partições\logradouros_particao14_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao14_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 15ok! Salvando partições\logradouros_particao15.gml
Executando emparelhamento...(39-25) : custo = 0.048
(61-76) : custo = 0.01

Custo adicional: 5.858
Salvando partições\logradouros_particao15_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao15_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 16ok! Salvando partições\logradouros_particao16.gml
Executando emparelhamento...(52-86) : custo = 0.036
(40-64) : custo = 0.014
(41-30) : custo = 0.008

Custo adicional: 5.858
Salvando partições\logradouros_particao16_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao16_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 17ok! Salvando partições\logradouros_particao17.gml
Executando emparelhamento...(257-276) : custo = 0.028
(273-234) : custo = 0.042
(231-245) : custo = 0.07
(221-205) : custo = 0.014
(244-218) : custo = 0.022
(220-219) : custo = 0.016

Custo adicional: 19.392
Salvando partições\logradouros_particao17_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao17_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 18ok! Salvando partições\logradouros_particao18.gml
Executando emparelhamento...(218-217) : custo = 0.016

Custo adicional: 1.616
Salvando partições\logradouros_particao18_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao18_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 19ok! Salvando partições\logradouros_particao19.gml
Executando emparelhamento...(88-89) : custo = 0.044

Custo adicional: 4.444
Salvando partições\logradouros_particao19_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao19_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 20ok! Salvando partições\logradouros_particao20.gml
Executando emparelhamento...
Custo adicional: 0
Salvando partições\logradouros_particao20_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao20_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 21ok! Salvando partições\logradouros_particao21.gml
Executando emparelhamento...(220-204) : custo = 0.014

Custo adicional: 1.414
Salvando partições\logradouros_particao21_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao21_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 22ok! Salvando partições\logradouros_particao22.gml
Executando emparelhamento...
Custo adicional: 0
Salvando partições\logradouros_particao22_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao22_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 23ok! Salvando partições\logradouros_particao23.gml
Executando emparelhamento...(65-66) : custo = 0.016
(88-90) : custo = 0.032
(41-43) : custo = 0.032
(7-11) : custo = 0.056

Custo adicional: 13.736
Salvando partições\logradouros_particao23_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao23_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 24ok! Salvando partições\logradouros_particao24.gml
Executando emparelhamento...(119-120) : custo = 0.018

Custo adicional: 1.818
Salvando partições\logradouros_particao24_emparelhado.gml
PERCURSO
Salvando caminho em partições\logradouros_particao24_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...
Extraindo particao 25ok! Salvando partições\logradouros_particao25.gml
Executando emparelhamento...
Custo adicional: 0
Salvando partições\logradouros_particao25_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao25_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 26ok! Salvando partições\logradouros_particao26.gml
Executando emparelhamento...(120-121) : custo = 0.018
(68-122) : custo = 0.038

Custo adicional: 5.656
Salvando partições\logradouros_particao26_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao26_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 27ok! Salvando partições\logradouros_particao27.gml
Executando emparelhamento...Extraindo componentes conexas...4
4

Numero de componentes: 4
Criando Gc...
  criando nodos...
(206-238) : custo = 0.176
Solucao 0: custo 32.376
(206-238) : custo = 0.176
Solucao 1: custo 32.376
(206-238) : custo = 0.176
Solucao 2: custo 32.376
(206-238) : custo = 0.176
Solucao 3: custo 32.376
(206-238) : custo = 0.176
Solucao 4: custo 32.376
(206-238) : custo = 0.176
Solucao 5: custo 32.376
Melhor solucao: 0

Custo adicional: 32.376
Salvando partições\logradouros_particao27_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao27_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Extraindo particao 28ok! Salvando partições\logradouros_particao28.gml
Executando emparelhamento...Extraindo componentes conexas...4
4
Numero de componentes: 4
Criando Gc...
  criando nodos...
(122-95) : custo = 0.064
(123-207) : custo = 0.054
(238-209) : custo = 0.03
(169-125) : custo = 0.026
Solucao 0: custo 25.474
(122-94) : custo = 0.046
(123-207) : custo = 0.054
(238-209) : custo = 0.03
(168-125) : custo = 0.044
Solucao 1: custo 25.474
(122-95) : custo = 0.064
(123-207) : custo = 0.054
(238-209) : custo = 0.03
(169-125) : custo = 0.026
Solucao 2: custo 25.474
(122-94) : custo = 0.046
(123-207) : custo = 0.054

(238-209) : custo = 0.03
(168-125) : custo = 0.044
Solucao 3: custo 25.474
(122-95) : custo = 0.064
(123-207) : custo = 0.054
(238-209) : custo = 0.03
(169-125) : custo = 0.026
Solucao 4: custo 25.474
(122-94) : custo = 0.046
(123-207) : custo = 0.054
(238-209) : custo = 0.03
(168-125) : custo = 0.044
Solucao 5: custo 25.474
Melhor solucao: 0

Custo adicional: 25.474
Salvando partições\logradouros_particao28_emparelhado.gml

PERCURSO
Salvando caminho em partições\logradouros_particao28_roteiro.txtLendo arquivo de segmentos (segmentos.txt)...

Gerando relatorio de particoes emparelhado logradouros_relatorio_particoes.txt
Salvando grafo particionado emparelhado logradouros_particionado_emparelhado.gml
Custo adicional total 191.17

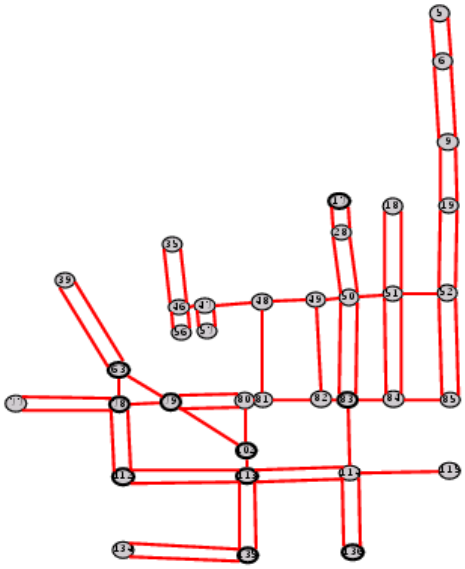
Part	Arcos	Custo	Vertices
1	50	243.7	25
2	66	246.9	38
3	20	237.9	11
4	11	237.2	7
5	24	243.8	12
6	23	245.7	13
7	32	246.6	17
8	10	280.6	6
9	28	238.7	13
10	14	234.9	8
11	31	252	15
12	16	242.1	8
13	14	242.6	7
14	18	239.5	8
15	31	242.3	15
16	38	251.4	17
17	47	249.1	27
18	8	231.6	5
19	22	257.2	10
20	10	231.5	7
21	18	232	10
22	14	237.8	8
23	46	257	25
24	18	233.3	10
25	18	225	9
26	36	247	17
27	48	238.1	25
28	40	244.7	22
Tot.	751	6810.2	395*
Med.	26.8	243.22	14.1

Desv. 14.8 10.65 7.99
* vértices que compartilham partições são contados uma vez por partição
Tempo de execucao: 15s.
FIM
?
Entre com o mapa (.gml) rede
1-15 [1]
2-5 [1]
3-4 [1]
6-8 [1]
7-11 [1]
9-10 [1]
12-13 [1]
14-19 [1]
16-23 [1]
17-25 [1]
18-22 [1]
20-21 [1]
24-26 [1]
27-28 [1]
FIM

APÊNDICE E – Resultados Gerados pelo Modelo em sua Aplicação no Estudo de Caso

Primeiro cenário

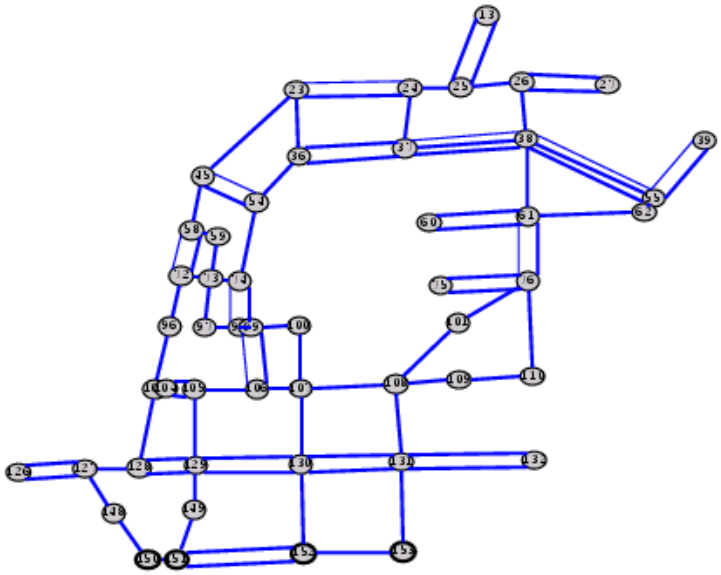
Partição 1



Roteiro partição 1
Tempo do percurso: 493.1min.

102 - 79 [uma passada]; 79 - 80 [lado par]; 80 - 81 [uma passada]; 81 - 80 [ocioso]; 80 - 79 [lado impar]; 79 - 78 [uma passada]; 78 - 77 [lado par]; 77 - 78 [lado impar]; 78 - 112 [lado par]; 112 - 78 [lado impar]; 78 - 63 [uma passada]; 63 - 39 [lado par]; 39 - 63 [lado impar]; 63 - 79 [uma passada]; 79 - 80 [ocioso]; 80 - 102 [uma passada]; 102 - 113 [uma passada]; 113 - 112 [lado par]; 112 - 113 [lado impar]; 113 - 135 [lado par]; 135 - 134 [lado par]; 134 - 135 [lado impar]; 135 - 113 [lado impar]; 113 - 114 [lado par]; 114 - 83 [uma passada]; 83 - 82 [uma passada]; 82 - 49 [uma passada]; 49 - 82 [ocioso]; 82 - 81 [uma passada]; 81 - 48 [uma passada]; 48 - 47 [uma passada]; 47 - 46 [uma passada]; 46 - 56 [lado par]; 56 - 46 [lado impar]; 46 - 35 [lado par]; 35 - 46 [lado impar]; 46 - 47 [ocioso]; 47 - 57 [lado par]; 57 - 47 [lado impar]; 47 - 48 [ocioso]; 48 - 49 [uma passada]; 49 - 50 [uma passada]; 50 - 83 [lado par]; 83 - 84 [uma passada]; 84 - 85 [uma passada]; 85 - 52 [lado par]; 52 - 85 [lado impar]; 85 - 52 [ocioso]; 52 - 19 [lado par]; 19 - 9 [lado par]; 9 - 6 [lado par]; 6 - 5 [lado par]; 5 - 6 [lado impar]; 6 - 9 [lado impar]; 9 - 19 [lado impar]; 19 - 52 [lado impar]; 52 - 51 [uma passada]; 51 - 84 [lado par]; 84 - 51 [lado impar]; 51 - 18 [lado par]; 18 - 51 [lado impar]; 51 - 50 [uma passada]; 50 - 28 [lado par]; 28 - 17 [lado par]; 17 - 28 [lado impar]; 28 - 50 [lado impar]; 50 - 83 [lado impar]; 83 - 114 [ocioso]; 114 - 136 [lado par]; 136 - 114 [lado impar]; 114 - 115 [uma passada]; 115 - 114 [ocioso]; 114 - 113 [lado impar]; 113 - 102 [ocioso]

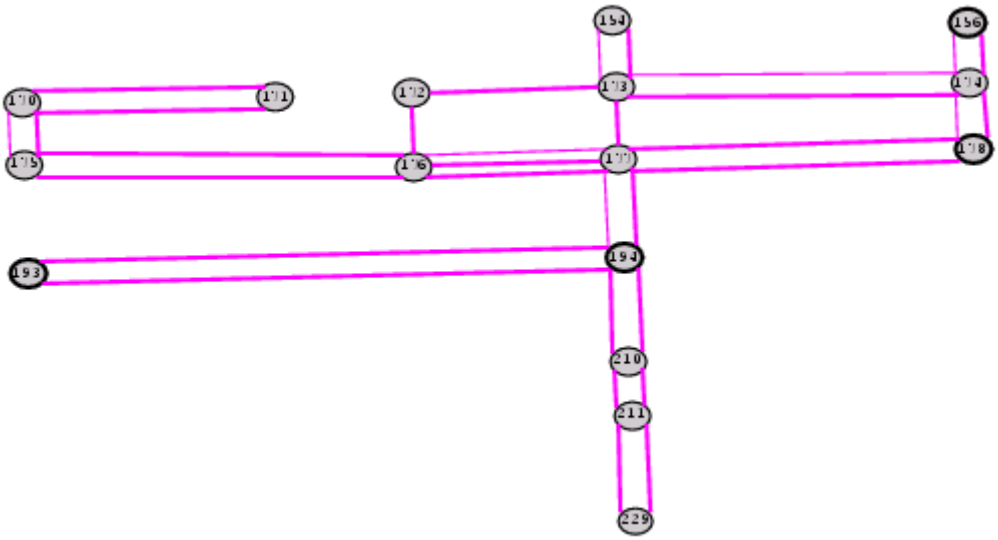
Partição 2



Roteiro partição 2
Tempo do percurso: 490.9min.

152 - 151 [lado par]; 151 - 150 [uma passada]; 150 - 148 [uma passada]; 148 - 127 [uma passada]; 127 - 126 [lado par]; 126 - 127 [lado impar]; 127 - 128 [uma passada]; 128 - 129 [lado par]; 129 - 128 [lado impar]; 128 - 103 [uma passada]; 103 - 96 [uma passada]; 96 - 72 [uma passada]; 72 - 58 [uma passada]; 58 - 72 [ocioso]; 72 - 73 [uma passada]; 73 - 74 [uma passada]; 74 - 98 [uma passada]; 98 - 97 [uma passada]; 97 - 73 [uma passada]; 73 - 59 [uma passada]; 59 - 58 [uma passada]; 58 - 45 [uma passada]; 45 - 54 [uma passada]; 54 - 45 [ocioso]; 45 - 23 [uma passada]; 23 - 36 [uma passada]; 36 - 54 [uma passada]; 54 - 74 [uma passada]; 74 - 98 [ocioso]; 98 - 99 [uma passada]; 99 - 106 [uma passada]; 106 - 99 [ocioso]; 99 - 100 [uma passada]; 100 - 107 [uma passada]; 107 - 106 [uma passada]; 106 - 105 [uma passada]; 105 - 104 [lado par]; 104 - 105 [lado impar]; 105 - 129 [uma passada]; 129 - 149 [uma passada]; 149 - 151 [uma passada]; 151 - 152 [lado impar]; 152 - 130 [uma passada]; 130 - 129 [lado par]; 129 - 130 [lado impar]; 130 - 107 [uma passada]; 107 - 108 [uma passada]; 108 - 101 [uma passada]; 101 - 76 [uma passada]; 76 - 61 [uma passada]; 61 - 60 [lado par]; 60 - 61 [lado impar]; 61 - 62 [uma passada]; 62 - 55 [uma passada]; 55 - 38 [lado par]; 38 - 37 [lado par]; 37 - 24 [uma passada]; 24 - 23 [uma passada]; 23 - 24 [ocioso]; 24 - 25 [uma passada]; 25 - 13 [lado par]; 13 - 25 [lado impar]; 25 - 26 [uma passada]; 26 - 27 [lado par]; 27 - 26 [lado impar]; 26 - 38 [uma passada]; 38 - 37 [lado impar]; 37 - 36 [lado par]; 36 - 37 [lado impar]; 37 - 38 [ocioso]; 38 - 55 [lado impar]; 55 - 39 [uma passada]; 39 - 55 [ocioso]; 55 - 38 [ocioso]; 38 - 61 [uma passada]; 61 - 76 [ocioso]; 76 - 75 [lado par]; 75 - 76 [lado impar]; 76 - 110 [uma passada]; 110 - 109 [uma passada]; 109 - 108 [uma passada]; 108 - 131 [uma passada]; 131 - 130 [lado par]; 130 - 131 [lado impar]; 131 - 132 [lado par]; 132 - 131 [lado impar]; 131 - 153 [uma passada]; 153 - 152 [uma passada]

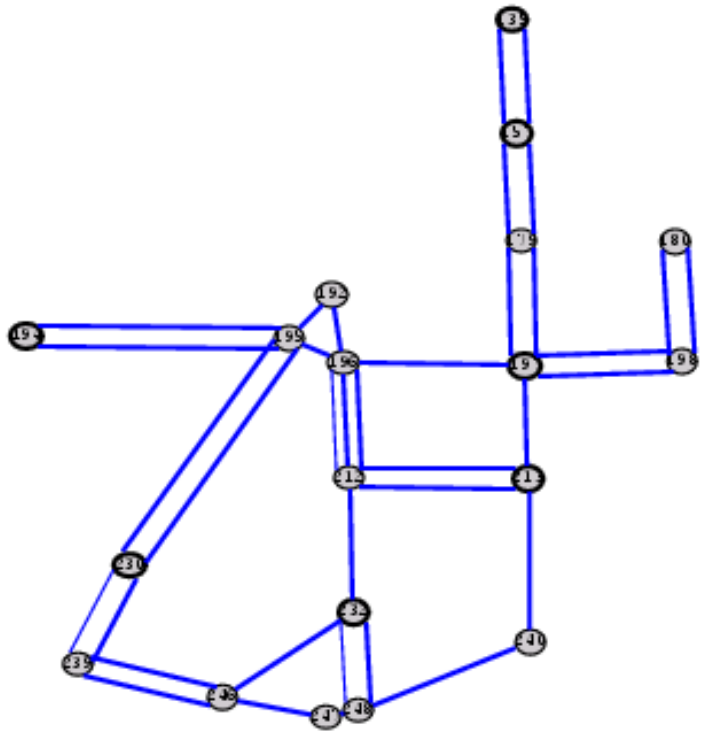
Partição 3



Roteiro partição 3
Tempo do percurso: 485.7min.

194 - 177 [uma passada]; 177 - 176 [lado par]; 176 - 175 [lado par]; 175 - 170 [uma passada]; 170 - 171 [lado par]; 171 - 170 [lado impar]; 170 - 175 [ocioso]; 175 - 176 [lado impar]; 176 - 172 [uma passada]; 172 - 173 [uma passada]; 173 - 174 [uma passada]; 174 - 178 [uma passada]; 178 - 177 [lado par]; 177 - 176 [lado impar]; 176 - 177 [ocioso]; 177 - 178 [lado impar]; 178 - 174 [ocioso]; 174 - 156 [uma passada]; 156 - 174 [ocioso]; 174 - 173 [ocioso]; 173 - 154 [uma passada]; 154 - 173 [ocioso]; 173 - 177 [uma passada]; 177 - 194 [ocioso]; 194 - 193 [lado par]; 193 - 194 [lado impar]; 194 - 210 [lado par]; 210 - 211 [lado par]; 211 - 229 [lado par]; 229 - 211 [lado impar]; 211 - 210 [lado impar]; 210 - 194 [lado impar]

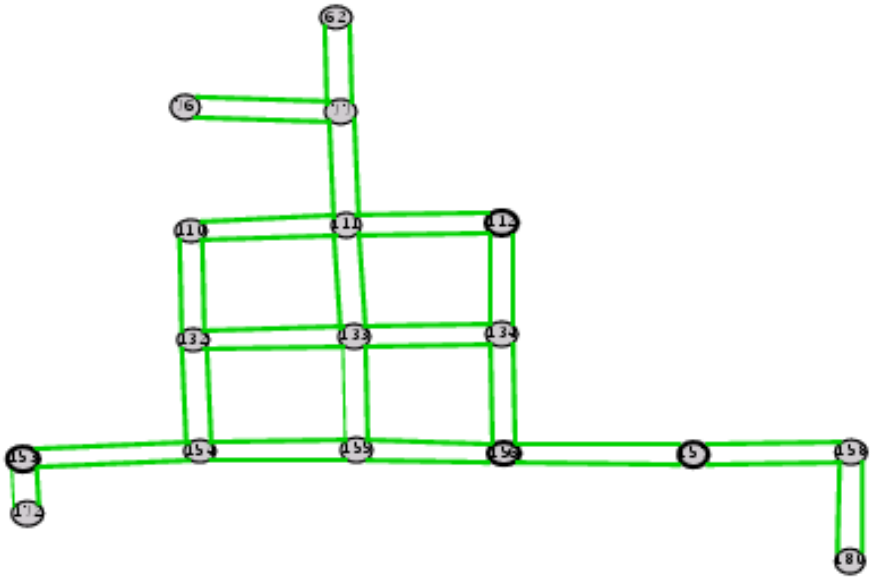
Partição 4



Roteiro partição 4
Tempo do percurso: 481.2min.

135 - 157 [lado par]; 157 - 179 [lado par]; 179 - 197 [lado par]; 197 - 213 [uma passada]; 213 - 240 [uma passada]; 240 - 248 [uma passada]; 248 - 232 [uma passada]; 232 - 248 [ocioso]; 248 - 247 [uma passada]; 247 - 246 [uma passada]; 246 - 239 [lado par]; 239 - 230 [uma passada]; 230 - 239 [ocioso]; 239 - 246 [lado impar]; 246 - 232 [uma passada]; 232 - 212 [uma passada]; 212 - 196 [lado par]; 196 - 212 [lado impar]; 212 - 213 [lado par]; 213 - 212 [lado impar]; 212 - 196 [ocioso]; 196 - 195 [uma passada]; 195 - 230 [lado par]; 230 - 195 [lado impar]; 195 - 194 [lado par]; 194 - 195 [lado impar]; 195 - 192 [uma passada]; 192 - 196 [uma passada]; 196 - 197 [uma passada]; 197 - 198 [lado par]; 198 - 180 [lado par]; 180 - 198 [lado impar]; 198 - 197 [lado impar]; 197 - 179 [lado impar]; 179 - 157 [lado impar]; 157 - 135 [lado impar]

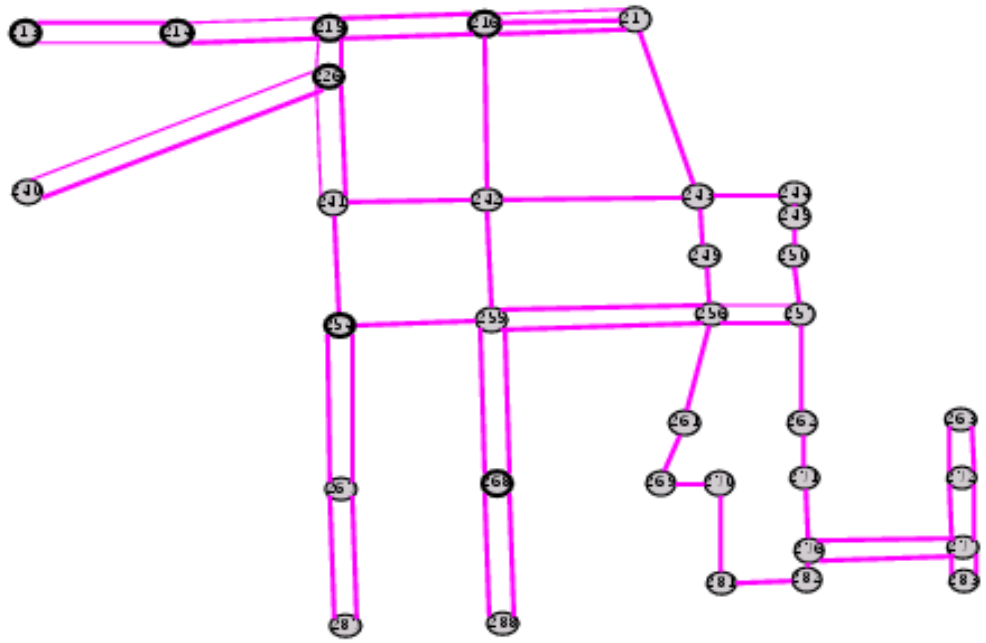
Partição 5



Roteiro partição 5
Tempo do percurso: 478.4min.

153 - 172 [uma passada]; 172 - 153 [ocioso]; 153 - 154 [lado par]; 154 - 132 [lado par]; 132 - 154 [lado impar]; 154 - 155 [lado par]; 155 - 133 [uma passada]; 133 - 132 [lado par]; 132 - 133 [lado impar]; 133 - 155 [ocioso]; 155 - 156 [lado par]; 156 - 134 [lado par]; 134 - 133 [lado par]; 133 - 134 [lado impar]; 134 - 112 [lado par]; 112 - 111 [lado par]; 111 - 110 [lado par]; 110 - 132 [lado par]; 132 - 110 [lado impar]; 110 - 111 [lado impar]; 111 - 133 [lado par]; 133 - 111 [lado impar]; 111 - 77 [lado par]; 77 - 76 [lado par]; 76 - 77 [lado impar]; 77 - 62 [lado par]; 62 - 77 [lado impar]; 77 - 111 [lado impar]; 111 - 112 [lado impar]; 112 - 134 [lado impar]; 134 - 156 [lado impar]; 156 - 157 [lado par]; 157 - 158 [lado par]; 158 - 180 [lado par]; 180 - 158 [lado impar]; 158 - 157 [lado impar]; 157 - 156 [lado impar]; 156 - 155 [lado impar]; 155 - 154 [lado impar]; 154 - 153 [lado impar]

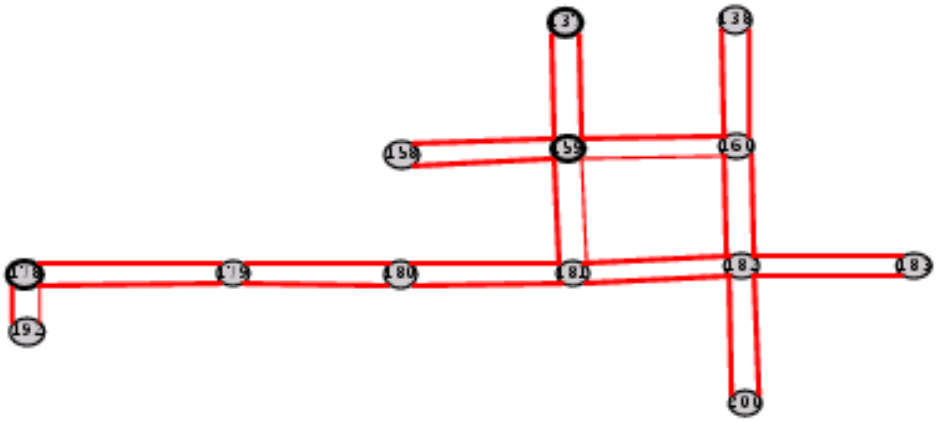
Partição 6



Roteiro partição 6
Tempo do percurso: 488.7min.

214 - 213 [uma passada]; 213 - 214 [ocioso]; 214 - 215 [uma passada]; 215 - 226 [uma passada]; 226 - 241 [uma passada]; 241 - 254 [uma passada]; 254 - 267 [lado par]; 267 - 287 [lado par]; 287 - 267 [lado impar]; 267 - 254 [lado impar]; 254 - 255 [uma passada]; 255 - 268 [lado par]; 268 - 288 [lado par]; 288 - 268 [lado impar]; 268 - 255 [lado impar]; 255 - 242 [uma passada]; 242 - 241 [uma passada]; 241 - 226 [ocioso]; 226 - 240 [uma passada]; 240 - 226 [ocioso]; 226 - 215 [ocioso]; 215 - 216 [lado par]; 216 - 242 [uma passada]; 242 - 243 [uma passada]; 243 - 249 [uma passada]; 249 - 256 [uma passada]; 256 - 261 [uma passada]; 261 - 269 [uma passada]; 269 - 270 [uma passada]; 270 - 281 [uma passada]; 281 - 282 [uma passada]; 282 - 276 [uma passada]; 276 - 277 [lado par]; 277 - 283 [lado par]; 283 - 277 [lado impar]; 277 - 272 [lado par]; 272 - 263 [lado par]; 263 - 272 [lado impar]; 272 - 277 [lado impar]; 277 - 276 [lado impar]; 276 - 271 [uma passada]; 271 - 262 [uma passada]; 262 - 257 [uma passada]; 257 - 256 [uma passada]; 256 - 255 [lado par]; 255 - 256 [lado impar]; 256 - 257 [ocioso]; 257 - 250 [uma passada]; 250 - 245 [uma passada]; 245 - 244 [uma passada]; 244 - 243 [uma passada]; 243 - 217 [uma passada]; 217 - 216 [lado par]; 216 - 217 [lado impar]; 217 - 216 [ocioso]; 216 - 215 [lado impar]; 215 - 214 [ocioso]

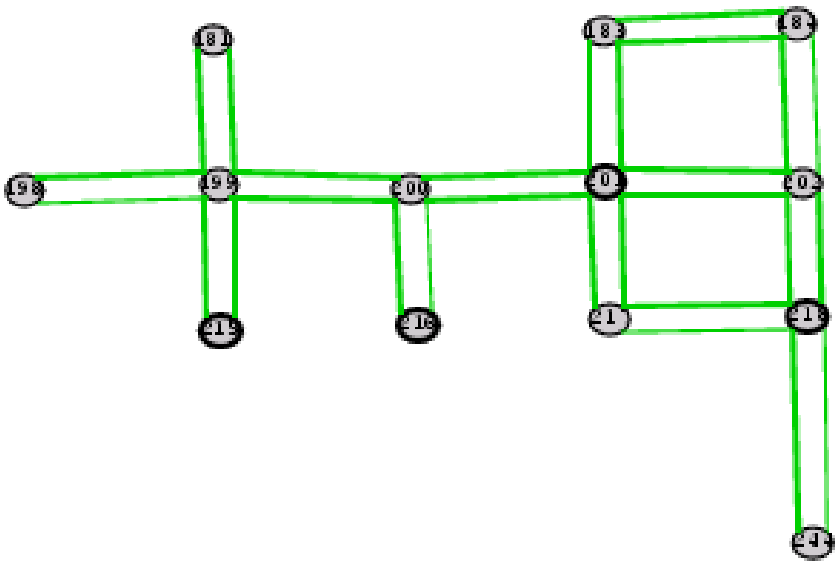
Partição 7



Roteiro partição 7
Tempo do percurso: 484.8min.

178 - 192 [uma passada]; 192 - 178 [ocioso]; 178 - 179 [lado par]; 179 - 180 [lado par]; 180 - 181 [lado par]; 181 - 159 [uma passada]; 159 - 158 [lado par]; 158 - 159 [lado impar]; 159 - 181 [ocioso]; 181 - 182 [lado par]; 182 - 200 [lado par]; 200 - 182 [lado impar]; 182 - 183 [lado par]; 183 - 182 [lado impar]; 182 - 160 [lado par]; 160 - 159 [uma passada]; 159 - 137 [lado par]; 137 - 159 [lado impar]; 159 - 160 [ocioso]; 160 - 138 [lado par]; 138 - 160 [lado impar]; 160 - 182 [lado impar]; 182 - 181 [lado impar]; 181 - 180 [lado impar]; 180 - 179 [lado impar]; 179 - 178 [lado impar]

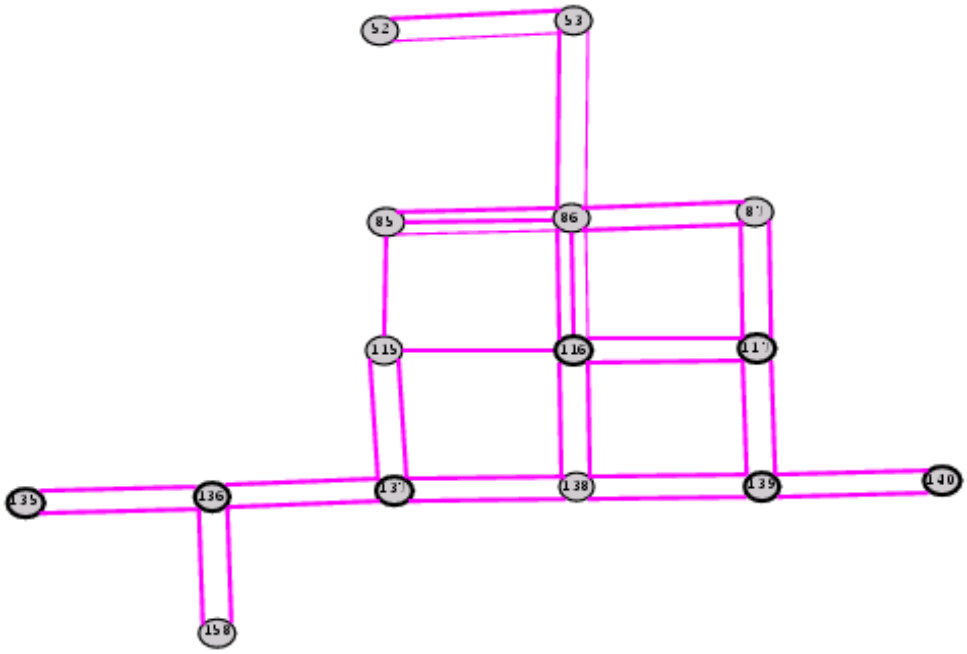
Partição 8



Roteiro partição 8
Tempo do percurso: 478.3min.

216 - 200 [uma passada]; 200 - 199 [lado par]; 199 - 215 [lado par]; 215 - 199 [lado impar]; 199 - 198 [uma passada]; 198 - 199 [ocioso]; 199 - 181 [lado par]; 181 - 199 [lado impar]; 199 - 200 [lado impar]; 200 - 201 [lado par]; 201 - 217 [lado par]; 217 - 218 [uma passada]; 218 - 244 [uma passada]; 244 - 218 [ocioso]; 218 - 217 [ocioso]; 217 - 201 [lado impar]; 201 - 183 [lado par]; 183 - 184 [lado par]; 184 - 202 [uma passada]; 202 - 218 [lado par]; 218 - 202 [lado impar]; 202 - 201 [lado par]; 201 - 183 [lado impar]; 183 - 184 [lado impar]; 184 - 202 [ocioso]; 202 - 201 [lado impar]; 201 - 200 [lado impar]; 200 - 216 [ocioso]

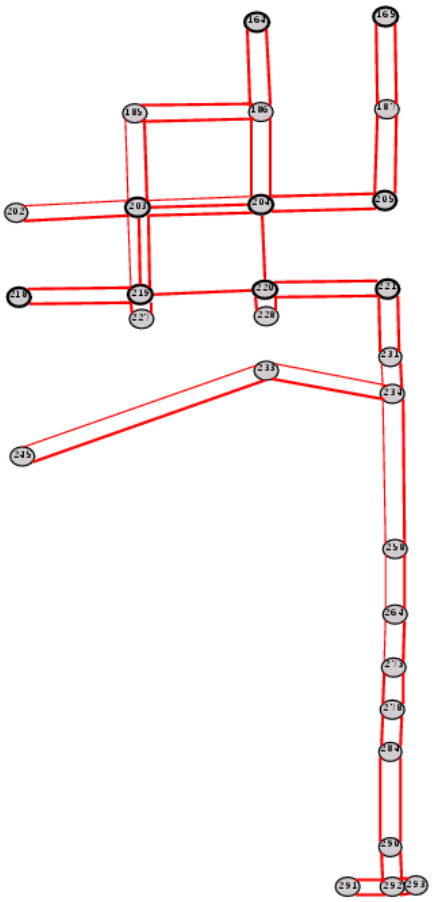
Partição 9



Roteiro partição 9
Tempo do percurso: 480min.

136 - 158 [lado par]; 158 - 136 [lado impar]; 136 - 135 [lado par]; 135 - 136 [lado impar]; 136 - 137 [lado par]; 137 - 115 [lado par]; 115 - 137 [lado impar]; 137 - 138 [lado par]; 138 - 139 [lado par]; 139 - 138 [lado impar]; 138 - 116 [lado par]; 116 - 86 [lado par]; 86 - 85 [lado par]; 85 - 115 [uma passada]; 115 - 116 [uma passada]; 116 - 117 [lado par]; 117 - 139 [lado par]; 139 - 140 [lado par]; 140 - 139 [lado impar]; 139 - 117 [lado impar]; 117 - 116 [lado impar]; 116 - 86 [lado impar]; 86 - 53 [uma passada]; 53 - 52 [uma passada]; 52 - 53 [ocioso]; 53 - 86 [ocioso]; 86 - 85 [lado impar]; 85 - 86 [ocioso]; 86 - 87 [lado par]; 87 - 117 [lado par]; 117 - 87 [lado impar]; 87 - 86 [lado impar]; 86 - 116 [ocioso]; 116 - 138 [lado impar]; 138 - 137 [lado impar]; 137 - 136 [lado impar]

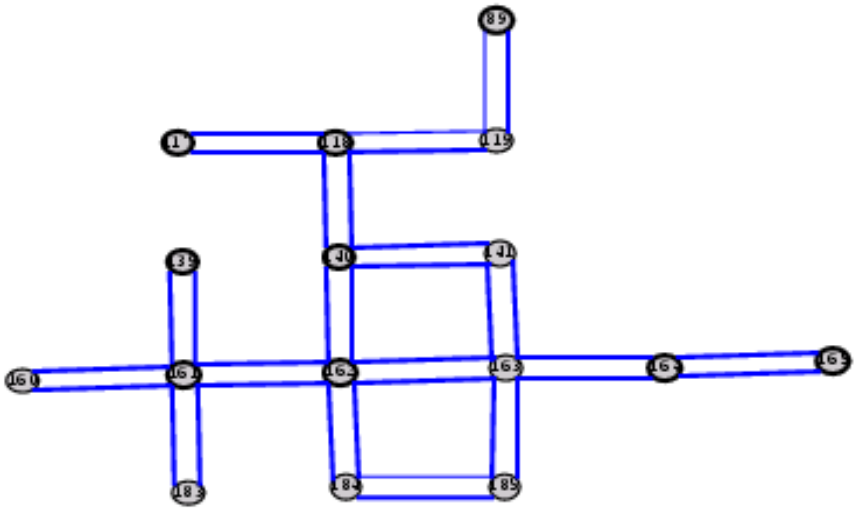
Partição 10



Roteiro partição 10
Tempo do percurso: 472.5min.

203 - 219 [lado par]; 219 - 227 [lado par]; 227 - 219 [lado impar]; 219 - 218 [lado par]; 218 - 219 [lado impar]; 219 - 203 [lado impar]; 203 - 202 [uma passada]; 202 - 203 [ocioso]; 203 - 219 [ocioso]; 219 - 220 [uma passada]; 220 - 228 [lado par]; 228 - 220 [lado impar]; 220 - 221 [lado par]; 221 - 231 [lado par]; 231 - 234 [uma passada]; 234 - 233 [uma passada]; 233 - 245 [uma passada]; 245 - 233 [ocioso]; 233 - 234 [ocioso]; 234 - 258 [uma passada]; 258 - 264 [uma passada]; 264 - 273 [uma passada]; 273 - 278 [lado par]; 278 - 284 [lado par]; 284 - 290 [lado par]; 290 - 292 [lado par]; 292 - 291 [lado par]; 291 - 292 [lado impar]; 292 - 293 [lado par]; 293 - 292 [lado impar]; 292 - 290 [lado impar]; 290 - 284 [lado impar]; 284 - 278 [lado impar]; 278 - 273 [lado impar]; 273 - 264 [ocioso]; 264 - 258 [ocioso]; 258 - 234 [ocioso]; 234 - 231 [ocioso]; 231 - 221 [lado impar]; 221 - 220 [lado impar]; 220 - 204 [uma passada]; 204 - 203 [lado par]; 203 - 185 [uma passada]; 185 - 203 [ocioso]; 203 - 204 [lado impar]; 204 - 186 [lado par]; 186 - 185 [lado par]; 185 - 186 [lado impar]; 186 - 164 [lado par]; 164 - 186 [lado impar]; 186 - 204 [lado impar]; 204 - 205 [lado par]; 205 - 187 [lado par]; 187 - 165 [lado par]; 165 - 187 [lado impar]; 187 - 205 [lado impar]; 205 - 204 [lado impar]; 204 - 203 [ocioso]

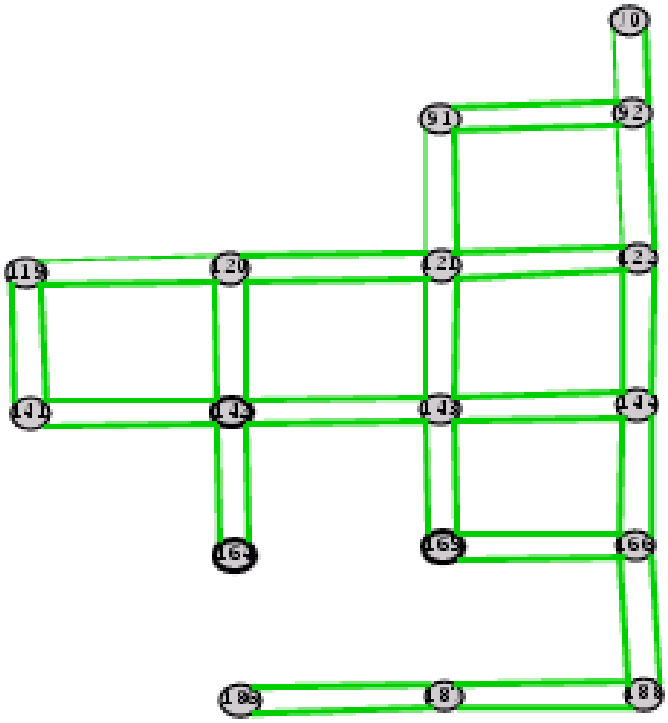
Partição 11



Roteiro partição 11
Tempo do percurso: 476.2min.

161 - 160 [lado par]; 160 - 161 [lado impar]; 161 - 183 [lado par]; 183 - 161 [lado impar]; 161 - 162 [lado par]; 162 - 184 [lado par]; 184 - 185 [uma passada]; 185 - 184 [ocioso]; 184 - 162 [lado impar]; 162 - 140 [lado par]; 140 - 162 [lado impar]; 162 - 163 [lado par]; 163 - 185 [lado par]; 185 - 163 [lado impar]; 163 - 141 [lado par]; 141 - 140 [lado par]; 140 - 118 [lado par]; 118 - 117 [lado par]; 117 - 118 [lado impar]; 118 - 119 [uma passada]; 119 - 89 [uma passada]; 89 - 119 [ocioso]; 119 - 118 [ocioso]; 118 - 140 [lado impar]; 140 - 141 [lado impar]; 141 - 163 [lado impar]; 163 - 164 [lado par]; 164 - 165 [lado par]; 165 - 164 [lado impar]; 164 - 163 [lado impar]; 163 - 162 [lado impar]; 162 - 161 [lado impar]; 161 - 139 [lado par]; 139 - 161 [lado impar]

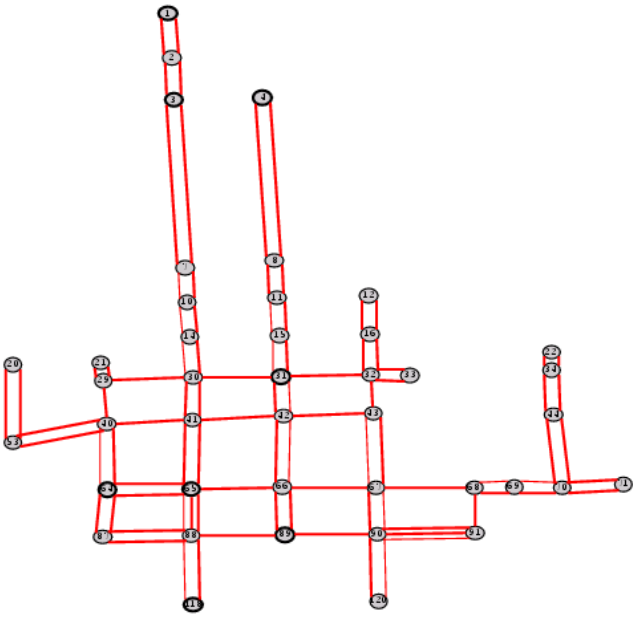
Partição 12



Roteiro partição 12
Tempo do percurso: 481.1min.

165 - 143 [lado par]; 143 - 142 [lado par]; 142 - 141 [lado par]; 141 - 119 [lado par]; 119 - 120 [uma passada]; 120 - 119 [ocioso]; 119 - 141 [lado impar]; 141 - 142 [lado impar]; 142 - 164 [lado par]; 164 - 142 [lado impar]; 142 - 120 [lado par]; 120 - 142 [lado impar]; 142 - 143 [lado impar]; 143 - 165 [lado impar]; 165 - 166 [lado par]; 166 - 188 [lado par]; 188 - 187 [lado par]; 187 - 186 [lado par]; 186 - 187 [lado impar]; 187 - 188 [lado impar]; 188 - 166 [lado impar]; 166 - 144 [lado par]; 144 - 143 [lado par]; 143 - 121 [lado par]; 121 - 120 [lado par]; 120 - 121 [lado impar]; 121 - 143 [lado impar]; 143 - 144 [lado impar]; 144 - 122 [lado par]; 122 - 121 [lado par]; 121 - 91 [uma passada]; 91 - 121 [ocioso]; 121 - 122 [lado impar]; 122 - 92 [uma passada]; 92 - 91 [lado par]; 91 - 92 [lado impar]; 92 - 70 [uma passada]; 70 - 92 [ocioso]; 92 - 122 [ocioso]; 122 - 144 [lado impar]; 144 - 166 [lado impar]; 166 - 165 [lado impar]

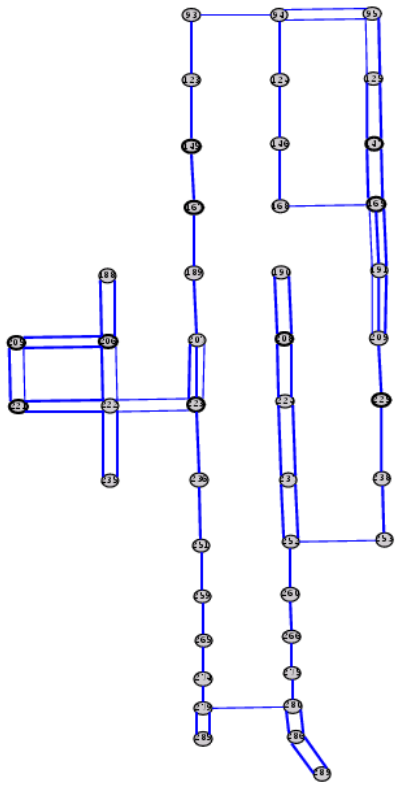
Partição 13



Roteiro partição 13
Tempo do percurso: 493.5min.

118 - 88 [uma passada]; 88 - 87 [lado par]; 87 - 64 [lado par]; 64 - 87 [lado impar]; 87 - 88 [lado impar]; 88 - 65 [lado par]; 65 - 64 [lado par]; 64 - 40 [uma passada]; 40 - 53 [lado par]; 53 - 20 [lado par]; 20 - 53 [lado impar]; 53 - 40 [lado impar]; 40 - 64 [ocioso]; 64 - 65 [lado impar]; 65 - 88 [lado impar]; 88 - 65 [ocioso]; 65 - 41 [lado par]; 41 - 40 [uma passada]; 40 - 29 [uma passada]; 29 - 21 [lado par]; 21 - 29 [lado impar]; 29 - 30 [uma passada]; 30 - 41 [lado par]; 41 - 65 [lado impar]; 65 - 66 [uma passada]; 66 - 89 [lado par]; 89 - 90 [uma passada]; 90 - 120 [uma passada]; 120 - 90 [ocioso]; 90 - 67 [uma passada]; 67 - 90 [ocioso]; 90 - 91 [lado par]; 91 - 90 [lado impar]; 90 - 91 [ocioso]; 91 - 68 [uma passada]; 68 - 69 [uma passada]; 69 - 70 [uma passada]; 70 - 44 [lado par]; 44 - 34 [lado par]; 34 - 22 [lado par]; 22 - 34 [lado impar]; 34 - 44 [lado impar]; 44 - 70 [lado impar]; 70 - 71 [lado par]; 71 - 70 [lado impar]; 70 - 69 [ocioso]; 69 - 68 [ocioso]; 68 - 67 [uma passada]; 67 - 66 [uma passada]; 66 - 42 [uma passada]; 42 - 41 [uma passada]; 41 - 30 [lado impar]; 30 - 14 [uma passada]; 14 - 10 [uma passada]; 10 - 7 [uma passada]; 7 - 3 [lado par]; 3 - 2 [lado par]; 2 - 1 [lado par]; 1 - 2 [lado impar]; 2 - 3 [lado impar]; 3 - 7 [lado impar]; 7 - 10 [ocioso]; 10 - 14 [ocioso]; 14 - 30 [ocioso]; 30 - 31 [uma passada]; 31 - 42 [uma passada]; 42 - 31 [ocioso]; 31 - 15 [uma passada]; 15 - 11 [uma passada]; 11 - 8 [lado par]; 8 - 4 [lado par]; 4 - 8 [lado impar]; 8 - 11 [lado impar]; 11 - 15 [ocioso]; 15 - 31 [ocioso]; 31 - 32 [uma passada]; 32 - 16 [lado par]; 16 - 12 [lado par]; 12 - 16 [lado impar]; 16 - 32 [lado impar]; 32 - 33 [lado par]; 33 - 32 [lado impar]; 32 - 43 [uma passada]; 43 - 67 [uma passada]; 67 - 43 [ocioso]; 43 - 42 [uma passada]; 42 - 66 [ocioso]; 66 - 89 [lado impar]; 89 - 88 [uma passada]; 88 - 118 [ocioso]

Partição 14

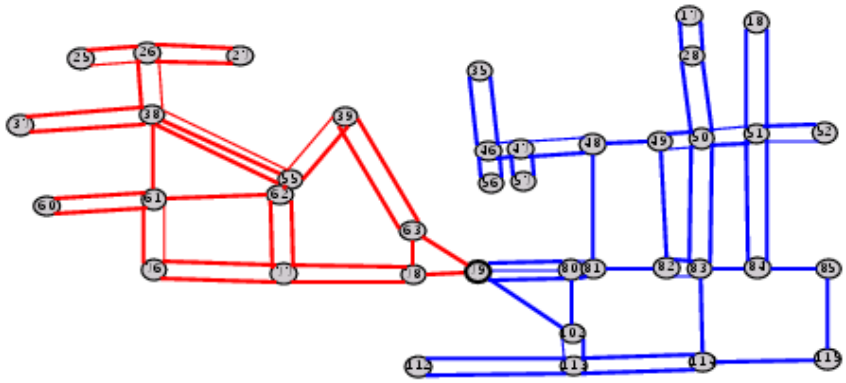


Roteiro partição 14
Tempo do percurso: 472.5min.

221 - 205 [uma passada]; 205 - 221 [ocioso]; 221 - 222 [lado par]; 222 - 206 [uma passada]; 206 - 205 [lado par]; 205 - 206 [lado impar]; 206 - 188 [lado par]; 188 - 206 [lado impar]; 206 - 222 [ocioso]; 222 - 223 [uma passada]; 223 - 236 [uma passada]; 236 - 251 [uma passada]; 251 - 259 [uma passada]; 259 - 265 [uma passada]; 265 - 274 [uma passada]; 274 - 279 [uma passada]; 279 - 285 [lado par]; 285 - 279 [lado impar]; 279 - 280 [uma passada]; 280 - 286 [lado par]; 286 - 289 [lado par]; 289 - 286 [lado impar]; 286 - 280 [lado impar]; 280 - 275 [uma passada]; 275 - 266 [uma passada]; 266 - 260 [uma passada]; 260 - 252 [uma passada]; 252 - 237 [lado par]; 237 - 224 [lado par]; 224 - 208 [lado par]; 208 - 190 [lado par]; 190 - 208 [lado impar]; 208 - 224 [lado impar]; 224 - 237 [lado impar]; 237 - 252 [lado impar]; 252 - 253 [uma passada]; 253 - 238 [uma passada]; 238 - 225 [uma passada]; 225 - 209 [uma passada]; 209 - 191 [lado par]; 191 - 209 [lado impar]; 209 - 191 [ocioso]; 191 - 169 [lado par]; 169 - 191 [lado impar]; 191 - 169 [ocioso]; 169 - 168 [uma passada]; 168 - 146 [uma passada]; 146 - 124 [uma passada]; 124 - 94 [uma passada]; 94 - 95 [uma passada]; 95 - 125 [uma passada]; 125 - 147 [lado par]; 147 - 169 [lado par]; 169 - 147 [lado impar]; 147 - 125 [lado impar]; 125 - 95 [ocioso]; 95 - 94 [ocioso]; 94 - 93 [uma passada]; 93 - 123 [uma passada]; 123 - 145 [uma passada]; 145 - 167 [uma passada]; 167 - 189 [uma passada]; 189 - 207 [uma passada]; 207 - 223 [lado par]; 223 - 207 [lado impar]; 207 - 223 [ocioso]; 223 - 222 [ocioso]; 222 - 235 [lado par]; 235 - 222 [lado impar]; 222 - 221 [lado impar]

Segundo cenário

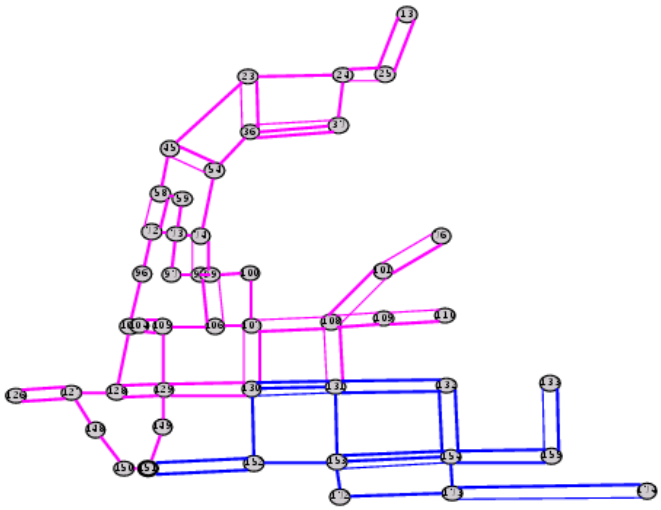
Partição 1



Roteiro partição agregada 1-15
Tempo do percurso: 243.7 + 242.3 = 486min.

79 - 80 [lado par]; 80 - 102 [uma passada]; 102 - 113 [uma passada]; 113 - 112 [lado par]; 112 - 113 [lado impar]; 113 - 102 [ocioso]; 102 - 79 [uma passada]; 79 - 80 [lado impar]; 80 - 81 [uma passada]; 81 - 48 [uma passada]; 48 - 47 [uma passada]; 47 - 57 [lado par]; 57 - 47 [lado impar]; 47 - 46 [uma passada]; 46 - 56 [lado par]; 56 - 46 [lado impar]; 46 - 35 [lado par]; 35 - 46 [lado impar]; 46 - 47 [ocioso]; 47 - 48 [ocioso]; 48 - 49 [uma passada]; 49 - 82 [uma passada]; 82 - 83 [uma passada]; 83 - 114 [uma passada]; 114 - 113 [lado par]; 113 - 114 [lado impar]; 114 - 115 [uma passada]; 115 - 85 [uma passada]; 85 - 84 [uma passada]; 84 - 83 [uma passada]; 83 - 50 [lado par]; 50 - 49 [uma passada]; 49 - 50 [ocioso]; 50 - 28 [lado par]; 28 - 17 [lado par]; 17 - 28 [lado impar]; 28 - 50 [lado impar]; 50 - 51 [uma passada]; 51 - 84 [lado par]; 84 - 51 [lado impar]; 51 - 18 [lado par]; 18 - 51 [lado impar]; 51 - 52 [uma passada]; 52 - 51 [ocioso]; 51 - 50 [ocioso]; 50 - 83 [lado impar]; 83 - 82 [ocioso]; 82 - 81 [uma passada]; 81 - 80 [ocioso]; 80 - 79 [ocioso];
79 - 78 [uma passada]; 78 - 77 [lado par]; 77 - 76 [lado par]; 76 - 61 [uma passada]; 61 - 60 [lado par]; 60 - 61 [lado impar]; 61 - 76 [ocioso]; 76 - 77 [lado impar]; 77 - 62 [lado par]; 62 - 77 [lado impar]; 77 - 78 [lado impar]; 78 - 63 [uma passada]; 63 - 39 [lado par]; 39 - 55 [uma passada]; 55 - 62 [uma passada]; 62 - 61 [uma passada]; 61 - 38 [uma passada]; 38 - 37 [lado par]; 37 - 38 [lado impar]; 38 - 55 [lado par]; 55 - 38 [lado impar]; 38 - 26 [uma passada]; 26 - 25 [uma passada]; 25 - 26 [ocioso]; 26 - 27 [lado par]; 27 - 26 [lado impar]; 26 - 38 [ocioso]; 38 - 55 [ocioso]; 55 - 39 [ocioso]; 39 - 63 [lado impar]; 63 - 79 [uma passada]

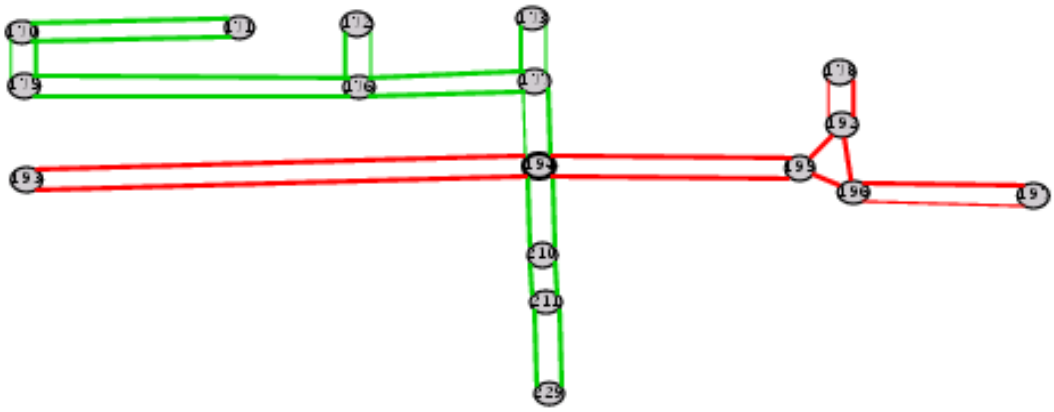
Partição 2



Roteiro partição agregada 2-5
Tempo do percurso: 246.9 + 243.8 = 490.7min.

151 - 149 [uma passada]; 149 - 129 [uma passada]; 129 - 105 [uma passada]; 105 - 104 [lado par]; 104 - 105 [lado impar]; 105 - 106 [uma passada]; 106 - 99 [uma passada]; 99 - 106 [ocioso]; 106 - 107 [uma passada]; 107 - 130 [uma passada]; 130 - 129 [lado par]; 129 - 128 [lado par]; 128 - 103 [uma passada]; 103 - 96 [uma passada]; 96 - 72 [uma passada]; 72 - 73 [uma passada]; 73 - 59 [uma passada]; 59 - 58 [uma passada]; 58 - 72 [uma passada]; 72 - 58 [ocioso]; 58 - 45 [uma passada]; 45 - 54 [uma passada]; 54 - 45 [ocioso]; 45 - 23 [uma passada]; 23 - 36 [uma passada]; 36 - 23 [ocioso]; 23 - 24 [uma passada]; 24 - 25 [uma passada]; 25 - 13 [lado par]; 13 - 25 [lado impar]; 25 - 24 [ocioso]; 24 - 37 [uma passada]; 37 - 36 [lado par]; 36 - 37 [lado impar]; 37 - 36 [ocioso]; 36 - 54 [uma passada]; 54 - 74 [uma passada]; 74 - 98 [uma passada]; 98 - 97 [uma passada]; 97 - 73 [uma passada]; 73 - 74 [uma passada]; 74 - 98 [ocioso]; 98 - 99 [uma passada]; 99 - 100 [uma passada]; 100 - 107 [uma passada]; 107 - 108 [uma passada]; 108 - 101 [uma passada]; 101 - 76 [uma passada]; 76 - 101 [ocioso]; 101 - 108 [ocioso]; 108 - 131 [uma passada]; 131 - 108 [ocioso]; 108 - 109 [uma passada]; 109 - 110 [uma passada]; 110 - 109 [ocioso]; 109 - 108 [ocioso]; 108 - 107 [ocioso]; 107 - 130 [ocioso]; 130 - 129 [lado impar]; 129 - 128 [lado impar]; 128 - 127 [uma passada]; 127 - 126 [lado par]; 126 - 127 [lado impar]; 127 - 148 [uma passada]; 148 - 150 [uma passada]; 150 - 151 [uma passada];
151 - 152 [lado par]; 152 - 130 [uma passada]; 130 - 131 [lado par]; 131 - 153 [uma passada]; 153 - 172 [uma passada]; 172 - 173 [uma passada]; 173 - 174 [uma passada]; 174 - 173 [ocioso]; 173 - 154 [uma passada]; 154 - 153 [lado par]; 153 - 154 [lado impar]; 154 - 155 [lado par]; 155 - 133 [uma passada]; 133 - 155 [ocioso]; 155 - 154 [lado impar]; 154 - 132 [lado par]; 132 - 131 [lado par]; 131 - 130 [lado impar]; 130 - 131 [ocioso]; 131 - 132 [lado impar]; 132 - 154 [lado impar]; 154 - 153 [ocioso]; 153 - 152 [uma passada]; 152 - 151 [lado impar]

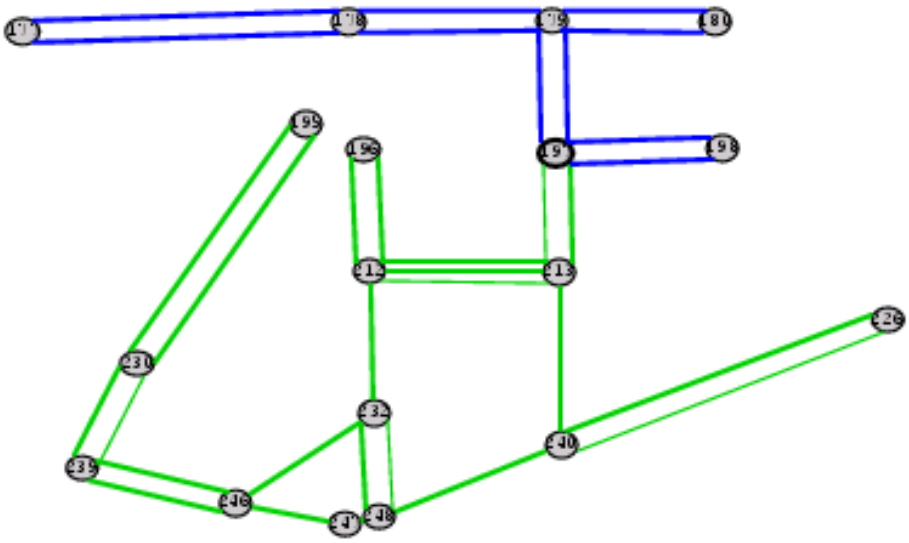
Partição 3



Roteiro partição agregada 3-4
Tempo do percurso: 237.9 + 237.2 = 475.1min.

194 - 210 [lado par]; 210 - 211 [lado par]; 211 - 229 [lado par]; 229 - 211 [lado impar]; 211 - 210 [lado impar]; 210 - 194 [lado impar]; 194 - 177 [uma passada]; 177 - 176 [lado par]; 176 - 175 [lado par]; 175 - 170 [uma passada]; 170 - 171 [lado par]; 171 - 170 [lado impar]; 170 - 175 [ocioso]; 175 - 176 [lado impar]; 176 - 172 [uma passada]; 172 - 176 [ocioso]; 176 - 177 [lado impar]; 177 - 173 [uma passada]; 173 - 177 [ocioso]; 177 - 194 [ocioso]; 194 - 193 [lado par]; 193 - 194 [lado impar]; 194 - 195 [lado par]; 195 - 192 [uma passada]; 192 - 178 [uma passada]; 178 - 192 [ocioso]; 192 - 196 [uma passada]; 196 - 197 [uma passada]; 197 - 196 [ocioso]; 196 - 195 [uma passada]; 195 - 194 [lado impar]

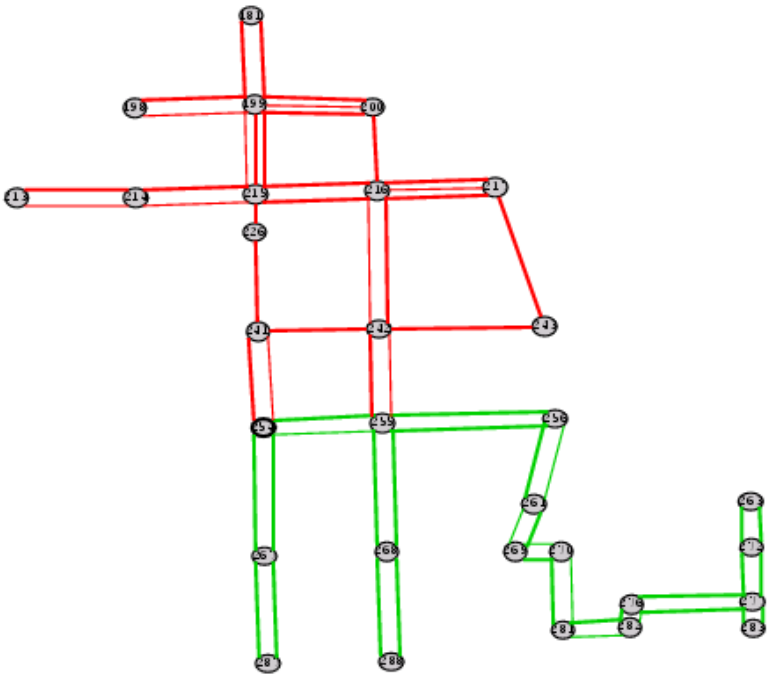
Partição 4



Roteiro partição agregada 6-8
Tempo do percurso: 245.7 + 280.6 = 526.3min.

197 - 213 [uma passada]; 213 - 240 [uma passada]; 240 - 226 [uma passada]; 226 - 240 [ocioso]; 240 - 248 [uma passada]; 248 - 247 [uma passada]; 247 - 246 [uma passada]; 246 - 239 [lado par]; 239 - 230 [uma passada]; 230 - 195 [lado par]; 195 - 230 [lado impar]; 230 - 239 [ocioso]; 239 - 246 [lado impar]; 246 - 232 [uma passada]; 232 - 248 [uma passada]; 248 - 232 [ocioso]; 232 - 212 [uma passada]; 212 - 213 [lado par]; 213 - 212 [lado impar]; 212 - 196 [lado par]; 196 - 212 [lado impar]; 212 - 213 [ocioso]; 213 - 197 [ocioso];
197 - 198 [lado par]; 198 - 197 [lado impar]; 197 - 179 [lado par]; 179 - 178 [lado par]; 178 - 177 [lado par]; 177 - 178 [lado impar]; 178 - 179 [lado impar]; 179 - 180 [lado par]; 180 - 179 [lado impar]; 179 - 197 [lado impar]

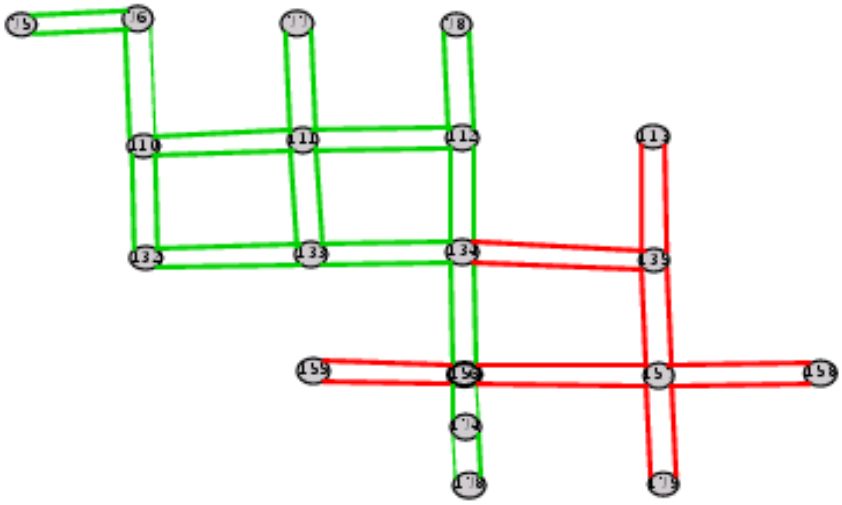
Partição 5



Roteiro partição agregada 7-11
Tempo do percurso: 246.6 + 252 = 498.6min.

254 - 267 [lado par]; 267 - 287 [lado par]; 287 - 267 [lado impar]; 267 - 254 [lado impar]; 254 - 255 [uma passada]; 255 - 268 [lado par]; 268 - 288 [lado par]; 288 - 268 [lado impar]; 268 - 255 [lado impar]; 255 - 256 [lado par]; 256 - 261 [uma passada]; 261 - 269 [uma passada]; 269 - 270 [uma passada]; 270 - 281 [uma passada]; 281 - 282 [uma passada]; 282 - 276 [uma passada]; 276 - 277 [lado par]; 277 - 283 [lado par]; 283 - 277 [lado impar]; 277 - 272 [lado par]; 272 - 263 [lado par]; 263 - 272 [lado impar]; 272 - 277 [lado impar]; 277 - 276 [lado impar]; 276 - 282 [ocioso]; 282 - 281 [ocioso]; 281 - 270 [ocioso]; 270 - 269 [ocioso]; 269 - 261 [ocioso]; 261 - 256 [ocioso]; 256 - 255 [lado impar]; 255 - 254 [ocioso]; 254 - 241 [uma passada]; 241 - 242 [uma passada]; 242 - 255 [uma passada]; 255 - 242 [ocioso]; 242 - 243 [uma passada]; 243 - 217 [uma passada]; 217 - 216 [lado par]; 216 - 215 [lado par]; 215 - 214 [uma passada]; 214 - 213 [uma passada]; 213 - 214 [ocioso]; 214 - 215 [ocioso]; 215 - 199 [lado par]; 199 - 198 [uma passada]; 198 - 199 [ocioso]; 199 - 215 [lado impar]; 215 - 199 [ocioso]; 199 - 181 [lado par]; 181 - 199 [lado impar]; 199 - 200 [lado par]; 200 - 199 [lado impar]; 199 - 200 [ocioso]; 200 - 216 [uma passada]; 216 - 242 [uma passada]; 242 - 216 [ocioso]; 216 - 217 [lado impar]; 217 - 216 [ocioso]; 216 - 215 [lado impar]; 215 - 226 [uma passada]; 226 - 241 [uma passada]; 241 - 254 [ocioso]

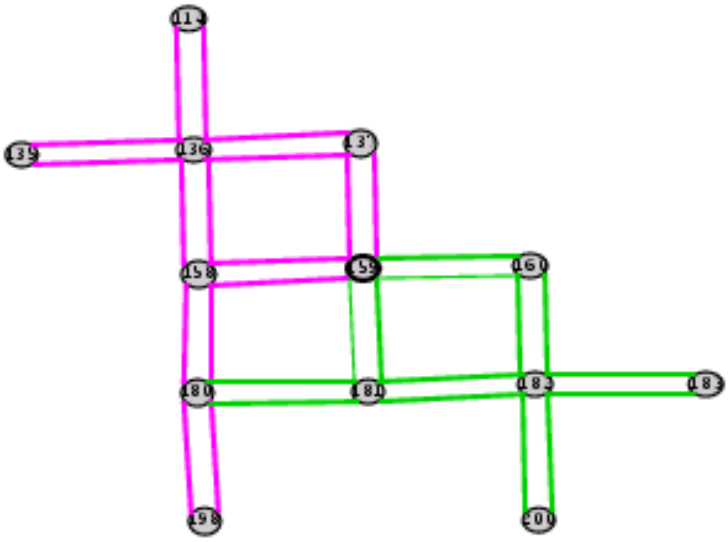
Partição 6



Roteiro partição agregada 9-10
Tempo do percurso: 238.7 + 234.9 = 473.6min.

156 - 174 [uma passada]; 174 - 178 [uma passada]; 178 - 174 [ocioso]; 174 - 156 [ocioso]; 156 - 134 [lado par]; 134 - 133 [lado par]; 133 - 132 [lado par]; 132 - 110 [lado par]; 110 - 132 [lado impar]; 132 - 133 [lado impar]; 133 - 111 [lado par]; 111 - 133 [lado impar]; 133 - 134 [lado impar]; 134 - 112 [lado par]; 112 - 111 [lado par]; 111 - 110 [lado par]; 110 - 76 [uma passada]; 76 - 75 [lado par]; 75 - 76 [lado impar]; 76 - 110 [ocioso]; 110 - 111 [lado impar]; 111 - 77 [lado par]; 77 - 111 [lado impar]; 111 - 112 [lado impar]; 112 - 78 [lado par]; 78 - 112 [lado impar]; 112 - 134 [lado impar]; 134 - 156 [lado impar];
156 - 155 [lado par]; 155 - 156 [lado impar]; 156 - 157 [lado par]; 157 - 179 [lado par]; 179 - 157 [lado impar]; 157 - 158 [lado par]; 158 - 157 [lado impar]; 157 - 135 [lado par]; 135 - 134 [lado par]; 134 - 135 [lado impar]; 135 - 113 [lado par]; 113 - 135 [lado impar]; 135 - 157 [lado impar]; 157 - 156 [lado impar]

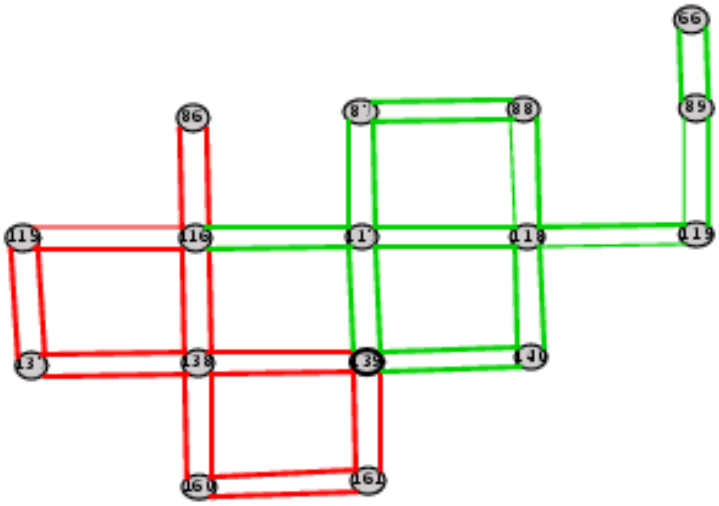
Partição 7



Roteiro partição agregada 12-13
Tempo do percurso: 242.1 + 242.6 = 484.7min.

159 - 158 [lado par]; 158 - 180 [lado par]; 180 - 198 [lado par]; 198 - 180 [lado impar]; 180 - 158 [lado impar]; 158 - 159 [lado impar]; 159 - 137 [lado par]; 137 - 136 [lado par]; 136 - 135 [lado par]; 135 - 136 [lado impar]; 136 - 158 [lado par]; 158 - 136 [lado impar]; 136 - 114 [lado par]; 114 - 136 [lado impar]; 136 - 137 [lado impar]; 137 - 159 [lado impar];
159 - 181 [uma passada]; 181 - 180 [lado par]; 180 - 181 [lado impar]; 181 - 182 [lado par]; 182 - 200 [lado par]; 200 - 182 [lado impar]; 182 - 181 [lado impar]; 181 - 159 [ocioso]; 159 - 160 [uma passada]; 160 - 182 [lado par]; 182 - 183 [lado par]; 183 - 182 [lado impar]; 182 - 160 [lado impar]; 160 - 159 [ocioso]

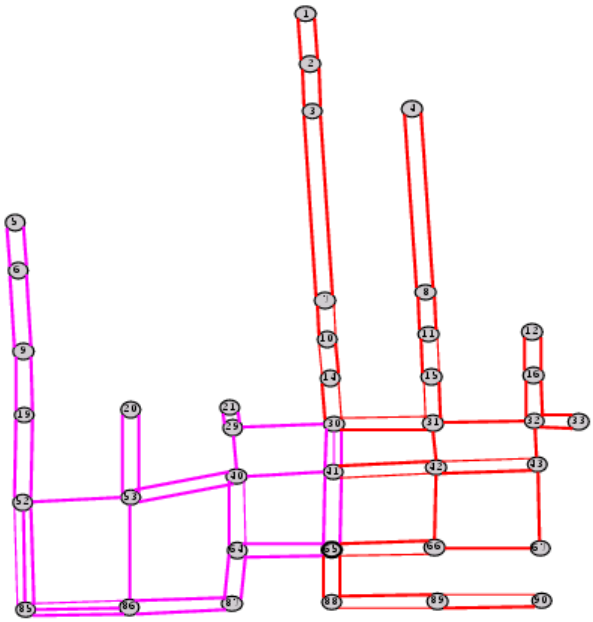
Partição 8



Roteiro partição agregada 14-19
Tempo do percurso: 239.5 + 257.2 = 496.7min.

139 - 161 [lado par]; 161 - 160 [lado par]; 160 - 161 [lado impar]; 161 - 139 [lado impar]; 139 - 138 [lado par]; 138 - 160 [lado par]; 160 - 138 [lado impar]; 138 - 137 [lado par]; 137 - 138 [lado impar]; 138 - 116 [lado par]; 116 - 115 [uma passada]; 115 - 137 [lado par]; 137 - 115 [lado impar]; 115 - 116 [ocioso]; 116 - 86 [lado par]; 86 - 116 [lado impar]; 116 - 138 [lado impar]; 138 - 139 [lado impar];
139 - 140 [lado par]; 140 - 139 [lado impar]; 139 - 117 [lado par]; 117 - 116 [lado par]; 116 - 117 [lado impar]; 117 - 87 [lado par]; 87 - 117 [lado impar]; 117 - 118 [lado par]; 118 - 140 [lado par]; 140 - 118 [lado impar]; 118 - 88 [uma passada]; 88 - 87 [lado par]; 87 - 88 [lado impar]; 88 - 118 [ocioso]; 118 - 119 [uma passada]; 119 - 89 [uma passada]; 89 - 66 [lado par]; 66 - 89 [lado impar]; 89 - 119 [ocioso]; 119 - 118 [ocioso]; 118 - 117 [lado impar]; 117 - 139 [lado impar]

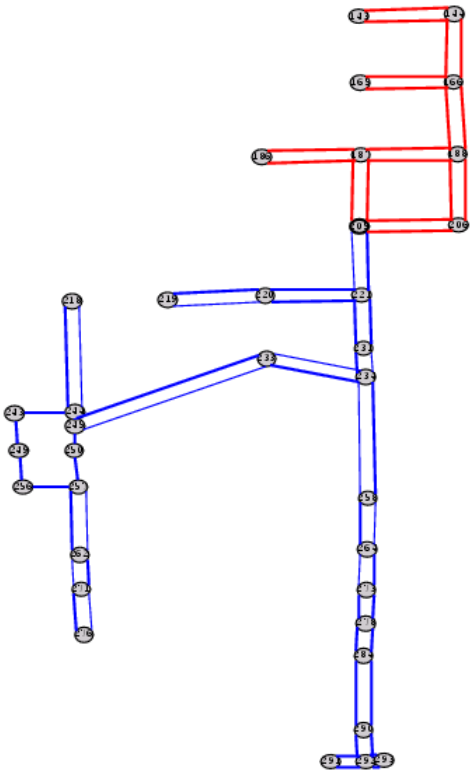
Partição 9



Roteiro partição agregada 16-23
Tempo do percurso: 251.4 + 257 = 508.4min.

65 - 64 [lado par]; 64 - 87 [lado par]; 87 - 86 [lado par]; 86 - 85 [lado par]; 85 - 52 [lado par]; 52 - 85 [lado impar]; 85 - 52 [ocioso]; 52 - 19 [lado par]; 19 - 9 [lado par]; 9 - 6 [lado par]; 6 - 5 [lado par]; 5 - 6 [lado impar]; 6 - 9 [lado impar]; 9 - 19 [lado impar]; 19 - 52 [lado impar]; 52 - 53 [uma passada]; 53 - 86 [uma passada]; 86 - 85 [lado impar]; 85 - 86 [ocioso]; 86 - 87 [lado impar]; 87 - 64 [lado impar]; 64 - 40 [uma passada]; 40 - 53 [lado par]; 53 - 20 [lado par]; 20 - 53 [lado impar]; 53 - 40 [lado impar]; 40 - 64 [ocioso]; 64 - 65 [lado impar]; 65 - 41 [lado par]; 41 - 40 [uma passada]; 40 - 29 [uma passada]; 29 - 21 [lado par]; 21 - 29 [lado impar]; 29 - 30 [uma passada]; 30 - 41 [lado par]; 41 - 30 [lado impar]; 30 - 41 [ocioso]; 41 - 65 [lado impar];
65 - 88 [lado par]; 88 - 89 [uma passada]; 89 - 90 [uma passada]; 90 - 89 [ocioso]; 89 - 88 [ocioso]; 88 - 65 [lado impar]; 65 - 66 [uma passada]; 66 - 42 [uma passada]; 42 - 41 [uma passada]; 41 - 42 [ocioso]; 42 - 31 [uma passada]; 31 - 30 [uma passada]; 30 - 14 [uma passada]; 14 - 10 [uma passada]; 10 - 7 [uma passada]; 7 - 3 [lado par]; 3 - 2 [lado par]; 2 - 1 [lado par]; 1 - 2 [lado impar]; 2 - 3 [lado impar]; 3 - 7 [lado impar]; 7 - 10 [ocioso]; 10 - 14 [ocioso]; 14 - 30 [ocioso]; 30 - 31 [ocioso]; 31 - 15 [uma passada]; 15 - 11 [uma passada]; 11 - 8 [lado par]; 8 - 4 [lado par]; 4 - 8 [lado impar]; 8 - 11 [lado impar]; 11 - 15 [ocioso]; 15 - 31 [ocioso]; 31 - 32 [uma passada]; 32 - 33 [lado par]; 33 - 32 [lado impar]; 32 - 16 [lado par]; 16 - 12 [lado par]; 12 - 16 [lado impar]; 16 - 32 [lado impar]; 32 - 43 [uma passada]; 43 - 42 [uma passada]; 42 - 43 [ocioso]; 43 - 67 [uma passada]; 67 - 66 [uma passada]; 66 - 65 [ocioso]

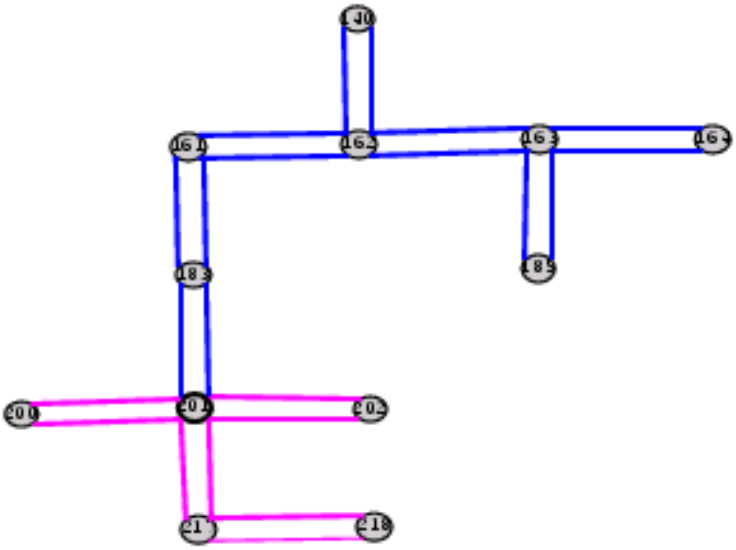
Partição 10



Roteiro partição agregada 17-25
Tempo do percurso: 249.1 + 225 = 474.1min.

205 - 221 [uma passada]; 221 - 231 [lado par]; 231 - 234 [uma passada]; 234 - 233 [uma passada]; 233 - 245 [uma passada]; 245 - 250 [uma passada]; 250 - 257 [uma passada]; 257 - 262 [uma passada]; 262 - 271 [uma passada]; 271 - 276 [uma passada]; 276 - 271 [ocioso]; 271 - 262 [ocioso]; 262 - 257 [ocioso]; 257 - 256 [uma passada]; 256 - 249 [uma passada]; 249 - 243 [uma passada]; 243 - 244 [uma passada]; 244 - 218 [uma passada]; 218 - 244 [ocioso]; 244 - 245 [uma passada]; 245 - 233 [ocioso]; 233 - 234 [ocioso]; 234 - 258 [uma passada]; 258 - 264 [uma passada]; 264 - 273 [uma passada]; 273 - 278 [lado par]; 278 - 284 [lado par]; 284 - 290 [lado par]; 290 - 292 [lado par]; 292 - 291 [lado par]; 291 - 292 [lado impar]; 292 - 293 [lado par]; 293 - 292 [lado impar]; 292 - 290 [lado impar]; 290 - 284 [lado impar]; 284 - 278 [lado impar]; 278 - 273 [lado impar]; 273 - 264 [ocioso]; 264 - 258 [ocioso]; 258 - 234 [ocioso]; 234 - 231 [ocioso]; 231 - 221 [lado impar]; 221 - 220 [lado par]; 220 - 219 [uma passada]; 219 - 220 [ocioso]; 220 - 221 [lado impar]; 221 - 205 [ocioso];
205 - 187 [lado par]; 187 - 186 [lado par]; 186 - 187 [lado impar]; 187 - 205 [lado impar]; 205 - 206 [lado par]; 206 - 188 [lado par]; 188 - 187 [lado par]; 187 - 188 [lado impar]; 188 - 166 [lado par]; 166 - 165 [lado par]; 165 - 166 [lado impar]; 166 - 144 [lado par]; 144 - 143 [lado par]; 143 - 144 [lado impar]; 144 - 166 [lado impar]; 166 - 188 [lado impar]; 188 - 206 [lado impar]; 206 - 205 [lado impar]

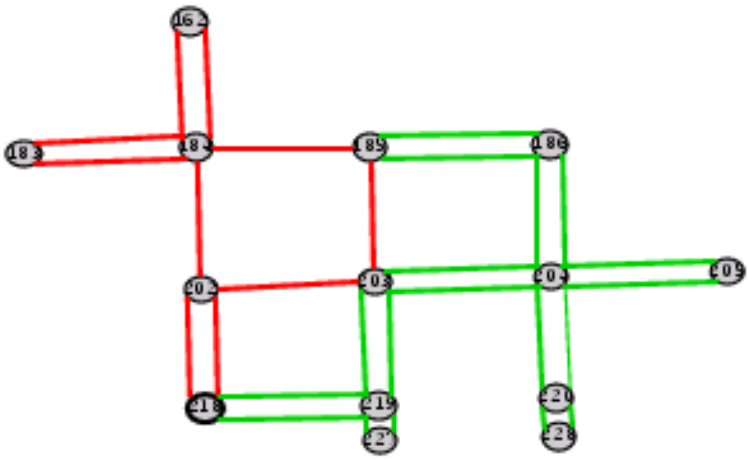
Partição 11



Roteiro partição agregada 18-22
Tempo do percurso: 231.6 + 237.8 = 469.4min.

201 - 200 [lado par]; 200 - 201 [lado impar]; 201 - 217 [lado par]; 217 - 218 [uma passada]; 218 - 217 [ocioso]; 217 - 201 [lado impar]; 201 - 202 [lado par]; 202 - 201 [lado impar]; 201 - 183 [lado par]; 183 - 161 [lado par]; 161 - 162 [lado par]; 162 - 163 [lado par]; 163 - 185 [lado par]; 185 - 163 [lado impar]; 163 - 164 [lado par]; 164 - 163 [lado impar]; 163 - 162 [lado impar]; 162 - 140 [lado par]; 140 - 162 [lado impar]; 162 - 161 [lado impar]; 161 - 183 [lado impar]; 183 - 201 [lado impar]

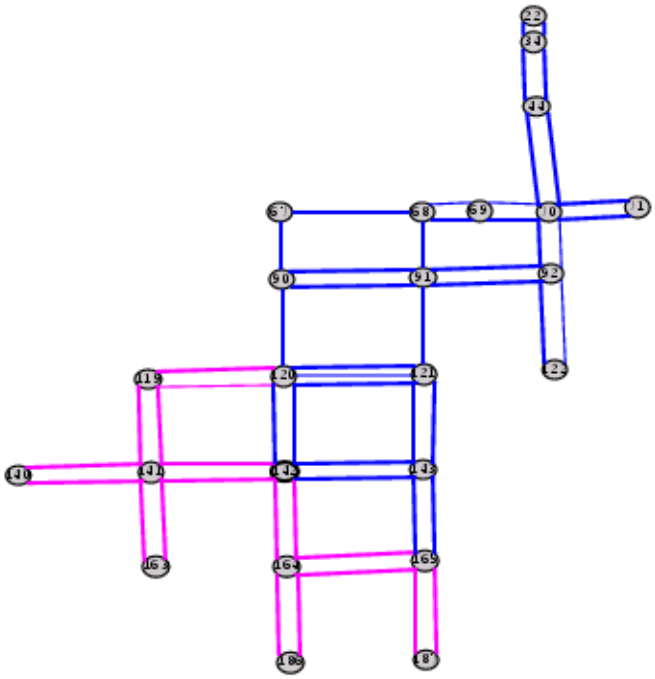
Partição 12



Roteiro partição agregada 20-21
Tempo do percurso: 231.5 + 232 = 463.5min.

218 - 202 [lado par]; 202 - 184 [uma passada]; 184 - 183 [lado par]; 183 - 184 [lado impar]; 184 - 162 [lado par]; 162 - 184 [lado impar]; 184 - 185 [uma passada]; 185 - 203 [uma passada]; 203 - 202 [uma passada]; 202 - 218 [lado impar];
218 - 219 [lado par]; 219 - 227 [lado par]; 227 - 219 [lado impar]; 219 - 203 [lado par]; 203 - 204 [lado par]; 204 - 220 [uma passada]; 220 - 228 [lado par]; 228 - 220 [lado impar]; 220 - 204 [ocioso]; 204 - 205 [lado par]; 205 - 204 [lado impar]; 204 - 186 [lado par]; 186 - 185 [lado par]; 185 - 186 [lado impar]; 186 - 204 [lado impar]; 204 - 203 [lado impar]; 203 - 219 [lado impar]; 219 - 218 [lado impar]

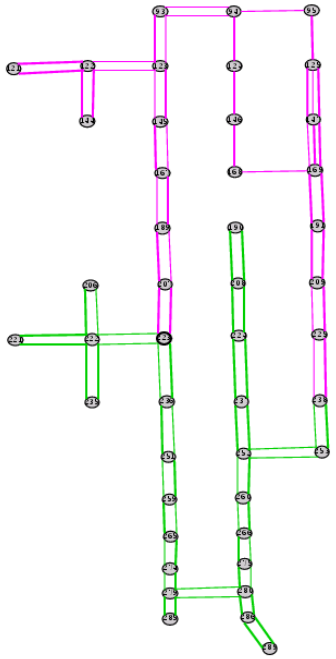
Partição 13



Roteiro partição agregada 24-26
Tempo do percurso: 233.3 + 247 = 480.3min.

142 - 164 [lado par]; 164 - 186 [lado par]; 186 - 164 [lado impar]; 164 - 165 [lado par]; 165 - 187 [lado par]; 187 - 165 [lado impar]; 165 - 164 [lado impar]; 164 - 142 [lado impar]; 142 - 141 [lado par]; 141 - 163 [lado par]; 163 - 141 [lado impar]; 141 - 140 [lado par]; 140 - 141 [lado impar]; 141 - 119 [lado par]; 119 - 120 [uma passada]; 120 - 119 [ocioso]; 119 - 141 [lado impar]; 141 - 142 [lado impar];
142 - 120 [lado par]; 120 - 142 [lado impar]; 142 - 143 [lado par]; 143 - 165 [lado par]; 165 - 143 [lado impar]; 143 - 121 [lado par]; 121 - 120 [lado par]; 120 - 90 [uma passada]; 90 - 67 [uma passada]; 67 - 68 [uma passada]; 68 - 91 [uma passada]; 91 - 90 [lado par]; 90 - 91 [lado impar]; 91 - 92 [lado par]; 92 - 122 [uma passada]; 122 - 92 [ocioso]; 92 - 70 [uma passada]; 70 - 69 [uma passada]; 69 - 68 [uma passada]; 68 - 69 [ocioso]; 69 - 70 [ocioso]; 70 - 71 [lado par]; 71 - 70 [lado impar]; 70 - 44 [lado par]; 44 - 34 [lado par]; 34 - 22 [lado par]; 22 - 34 [lado impar]; 34 - 44 [lado impar]; 44 - 70 [lado impar]; 70 - 92 [ocioso]; 92 - 91 [lado impar]; 91 - 121 [uma passada]; 121 - 120 [lado impar]; 120 - 121 [ocioso]; 121 - 143 [lado impar]; 143 - 142 [lado impar]

Partição 14



Roteiro partição agregada 27-28
Tempo do percurso: 238.1 + 244.7 = 482.8min.

223 - 236 [uma passada]; 236 - 251 [uma passada]; 251 - 259 [uma passada]; 259 - 265 [uma passada]; 265 - 274 [uma passada]; 274 - 279 [uma passada]; 279 - 285 [lado par]; 285 - 279 [lado impar]; 279 - 280 [uma passada]; 280 - 286 [lado par]; 286 - 289 [lado par]; 289 - 286 [lado impar]; 286 - 280 [lado impar]; 280 - 275 [uma passada]; 275 - 266 [uma passada]; 266 - 260 [uma passada]; 260 - 252 [uma passada]; 252 - 253 [uma passada]; 253 - 238 [uma passada]; 238 - 253 [ocioso]; 253 - 252 [ocioso]; 252 - 237 [lado par]; 237 - 224 [lado par]; 224 - 208 [lado par]; 208 - 190 [lado par]; 190 - 208 [lado impar]; 208 - 224 [lado impar]; 224 - 237 [lado impar]; 237 - 252 [lado impar]; 252 - 260 [ocioso]; 260 - 266 [ocioso]; 266 - 275 [ocioso]; 275 - 280 [ocioso]; 280 - 279 [ocioso]; 279 - 274 [ocioso]; 274 - 265 [ocioso]; 265 - 259 [ocioso]; 259 - 251 [ocioso]; 251 - 236 [ocioso]; 236 - 223 [ocioso]; 223 - 222 [uma passada]; 222 - 235 [lado par]; 235 - 222 [lado impar]; 222 - 206 [uma passada]; 206 - 222 [ocioso]; 222 - 221 [lado par]; 221 - 222 [lado impar]; 222 - 223 [ocioso];
223 - 207 [lado par]; 207 - 189 [uma passada]; 189 - 167 [uma passada]; 167 - 145 [uma passada]; 145 - 123 [uma passada]; 123 - 122 [uma passada]; 122 - 144 [lado par]; 144 - 122 [lado impar]; 122 - 121 [lado par]; 121 - 122 [lado impar]; 122 - 123 [ocioso]; 123 - 93 [uma passada]; 93 - 94 [uma passada]; 94 - 124 [uma passada]; 124 - 146 [uma passada]; 146 - 168 [uma passada]; 168 - 169 [uma passada]; 169 - 191 [lado par]; 191 - 209 [lado par]; 209 - 225 [uma passada]; 225 - 238 [uma passada]; 238 - 225 [ocioso]; 225 - 209 [ocioso]; 209 - 191 [lado impar]; 191 - 169 [lado impar]; 169 - 147 [lado par]; 147 - 169 [lado impar]; 169 - 147 [ocioso]; 147 - 125 [lado par]; 125 - 147 [lado impar]; 147 - 125 [ocioso]; 125 - 95 [uma passada]; 95 - 94 [uma passada]; 94 - 93 [ocioso]; 93 - 123 [ocioso]; 123 - 145 [ocioso]; 145 - 167 [ocioso]; 167 - 189 [ocioso]; 189 - 207 [ocioso]; 207 - 223 [lado impar]

ANEXO A – Algoritmo de Dijkstra para Determinação de um Caminho Mínimo

Inicialização

Para todo vértice $v_i \in V$ faça
Comece
dist (v_i) ← ∞;
final (v_i) ← *falso*;
pred (v_i) ← -1
Fim
dist (s) ← 0;
final (s) ← *verdadeiro*;
recente ← s;
[o vértice s é permanentemente rotulado com 0. Todos os outros vértices são temporariamente rotulados com ∞. O vértice s é o mais recente vértice a ser permanentemente rotulado]

Interação

Enquanto *final* (t) = *falso* faça
Comece
Para cada sucessor imediato v_i de *recente* que não seja *final*(v_i) faça
Comece [atualização dos rótulos temporários]
novo_rotulo ← *dist* (*recente*) + $w_{recente,\ vi}$
Se *novo_rotulo* < *dist* (v_i) então
Comece
dist (v_i) ← *novo_rotulo*;
pred (v_i) ← *recente*;
Fim
[rotule novamente v_i se existe um caminho mínimo através do vértice *recente* e torne *recente* o predecessor de v_i no caminho mínimo com início em s]
Fim
Seja y o vértice com o menor rótulo temporário cujo valor é diferente de ∞;
final (y) ← *verdadeiro*;
recente ← y;
[y, o vértice seguinte mais próximo de s torna-se permanentemente rotulado]
Fim

ANEXO B – Algoritmo de Gabow para Determinação de um Emparelhamento Máximo (todo emparelhamento perfeito é máximo)

E. Construa um emparelhamento máximo em um grafo. Comece uma busca por um caminho aumentante a cada vértice não saturado u. Explore arestas do grafo, decidindo designar novos rótulos ou aumentar o emparelhamento.

E0. [Inicialização] Leia o grafo contido em listas de adjacências, numerando os vértices de 1 a V e as arestas de $V + 1$ a $V + 2W$. Crie um vértice fictício O. Para $O \leq i \leq V$, faça *LABEL* $(i) \leftarrow -1$, *MATE* $(i) \leftarrow O$ (todos os vértices são não exteriores e não saturados). Faça $u \leftarrow O$.

E1. [Busca por um vértice não saturado] Faça $u \leftarrow u + 1$. Se $u > V$, pare; *MATE* contém um emparelhamento máximo. Caso contrário, se o vértice u é saturado, repita o passo *E1*. Caso contrário (u é não saturado, então designe um rótulo de início e comece uma nova busca). Faça *LABEL* $(u) \leftarrow$ *FIRST* $(u) \leftarrow O$.

E2. [Escolha de uma aresta] Escolha uma aresta xy, onde x é um vértice exterior. Uma aresta vw pode ser escolhida duas vezes em uma busca – uma vez com $x = v$, e outra vez com $x = w$. Se não existe tal aresta, vá para o passo *E7* (arestas xy podem ser escolhidas de modo arbitrário. Um método de escolha que pode ser utilizado é o *breadth-first*: um vértice exterior $x = x_1$ é escolhido, e arestas x_1y são escolhidas em sucessivas execuções do passo *E2*, quando todas estas arestas tiverem sido escolhidas, o vértice x_2 que estava rotulado imediatamente após x_1 é escolhido, e o processo é repetido para $x = x_2$. Este método *breadth-first* requer que o passo *E* mantenha uma lista de vértices exteriores, $x_1, x_2, ..., x_n$)

E3 [Expansão do emparelhamento] Se y é não saturado e $y \neq u$, faça *MATE* $(y) \leftarrow x$, chame R (x, y), então vá para *E7* (R completa a expansão ao longo do caminho $(y)^*P(x)$).

E4. [Designação dos rótulos de arestas] Se y é um vértice exterior, chame L, então vá para o passo *E2* (designa o rótulo de aresta $n(xy)$ para vértices não exteriores em $P(x)$ e $P(y)$).

E5.[Designação de um rótulo de vértice] Faça $v \leftarrow$ *MATE* (y). Se v é não exterior, faça *LABEL*$(v) \leftarrow x$, *FIRST* $(v) \leftarrow y$, e vá para o passo *E2*.

E6. [Tomar a próxima aresta] Vá para o passo *E2* (y é não exterior e *MATE* (y) é exterior, então a aresta xy nada adiciona).

E7. [Interrupção da busca] Faça *LABEL* $(O) \leftarrow -1$. Para todos os vértices exteriores i, faça *LABEL* $(i) \leftarrow$ *LABEL*(*MATE*$(i)) \leftarrow -1$. Então vá para *E1* (agora todos os vértices são não exteriores para a próxima busca).

L. Designe o rótulo de aresta $n(xy)$ para vértices não exteriores. A aresta xy liga os vértices x e y. *L* estabelece a *ligação* ao primeiro vértice em $P(x)$ e $P(y)$. Este rotula então todos os vértices não exteriores precedentes ligados a $P(x)$ ou $P(y)$.

L0. [Inicialização] Faça $r \leftarrow FIRST\ (x)$, $s \leftarrow FIRST\ (y)$. Se $r = s$, retorne (nenhum vértice pode ser rotulado). Caso contrário marque r e s. (os passos *L1 – L2* buscam *ligações* avançando alternadamente ao longo dos caminhos $P(x)$ e $P(y)$. Rótulos são designados aos vértices não exteriores nestes caminhos. Isso é feito igualando $LABEL\ (r)$ a um número negativo de aresta, $LABEL\ (r) \leftarrow - n(xy)$. Desse modo, cada invocação de *L* usa um rótulo de valor distinto).

L1. [Interrupção de caminhos] Se $s \neq O$, permute r e s, $r \leftrightarrow s$ (r é um vértice não exterior, alternadamente em $P(x)$ e $P(y)$).

L2. [Próximo vértice não exterior] Faça $r \leftarrow FIRST\ (LABEL(MATE(r)))$ (r é colocado no próximo vértice em $P(x)$ ou $P(y)$). Se r não está rotulado, rotule r e vá para o passo *L1*. Caso contrário, faça *ligação* $\leftarrow r$ e vá para o passo *L3*.

L3. [Rotular os vértices em $P(x)$ e $P(y)$] (todos os vértices não exteriores entre x e *ligação*, ou y e *ligação* serão designados rótulos de arestas). Faça $v \leftarrow FIRST\ (x)$ e execute o passo *L4*. Então faça $v \leftarrow FIRST\ (y)$ e execute o passo *L4*. Então vá para o passo *L5*.

L4. [Rotular v] Se $v \neq$ *ligação*, faça $LABEL(v) \leftarrow n\ (xy)$, $FIRST\ (v) \leftarrow$ *ligação*, $v \leftarrow FIRST(LABEL(MATE(v)))$ e repita o passo *L4*. Caso contrário continue como especificado no passo *L3*.

L5. [Atualizar *FIRST*] Para cada vértice exterior i, se $FIRST\ (i)$ é exterior, faça $FIRST\ (i) \leftarrow$ *ligação*. (*Ligação* é agora o primeiro vértice não exterior em $P(i)$)

L6. Retorne

$R(v, w)$ re-emparelham arestas no caminho aumentante. O vértice v é exterior. Parte do caminho $(w)^*P(v)$ está no caminho aumentante. Este se torna re-emparelhado por $R(v, w)$ (embora *R* faça $MATE(v) \leftarrow w$, este não faz $MATE(w) \leftarrow v$. Isto é feito no passo *E3* ou outro que chame por *R*). *R* é uma rotina recursiva.

R1. [Emparelhar v a w] Faça $i \leftarrow MATE\ (v)$, $MATE\ (v) \leftarrow w$. Se $MATE\ (i) \neq v$, retorne (o caminho está completamente re-emparelhado).

R2. [Re-emparelhar um caminho] Se v tem um rótulo de vértice, faça $MATE\ (i) \leftarrow LABEL\ (v)$, chame $R(LABEL(v), i)$ recursivamente, e então retorne.

R3. [Re-emparelhar dois caminhos] (o vértice v tem um rótulo de aresta) Estabeleça x, y como vértices, então $LABEL\ (v) = n(xy)$, chame $R(x, y)$ recursivamente, chame $R(y, x)$ recursivamente, e então retorne.

ANEXO C – Algoritmo de Kruskal para Determinação de uma Árvore de Peso Mínimo

Passo 1. Classifique as arestas do grafo pelo seu custo em ordem decrescente.

Passo 2. Examine todas as arestas, uma a uma, em ordem, da menor para a maior. Se uma aresta e_{ij} em análise é encontrada para forma um ciclo (quando adicionada as arestas já selecionadas), esta é descartada. Caso contrário, e_{ij} é selecionada para ser incluída na árvore de peso mínimo T. Repita até que todas as V - 1 arestas requeridas tenham sido selecionadas ou quando todas as arestas tiverem sido examinadas.