GUSTAVO PEIXOTO SILVA

UMA METODOLOGIA BASEADA NA TÉCNICA DE GERAÇÃO DE ARCOS PARA O PROBLEMA DE PROGRAMAÇÃO DE VEÍCULOS

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Doutor em Engenharia.

São Paulo

2001

Índice

Lista de Figuras

Lista de Tabelas

RESUMO

Este trabalho explora modelos de fluxo em redes para resolver de forma eficiente Problemas Práticos de Programação de Veículos no Sistema de Transporte Público (*Practical Mass Vehicle Scheduling Problem*). Duas técnicas de redução da rede com seus respectivos modelos de representação do problema são estudadas e testadas com dados reais.

A primeira metodologia combina o modelo de fluxo com custo mínimo à técnica de redução de arcos, denominada *Eliminação dos Arcos Longos*, que se baseia na proposta de Bokinge e Hasselström (1980) de excluir um determinado tipo de arcos da rede. Os arcos com altos custos são substituídos por outros arcos já existentes na rede e embora devam ser introduzidos novos arcos e nós auxiliares, ocorre uma redução considerável na dimensão total da rede. Foi explorada uma versão desta técnica, proposta por Freling et al. (1995), que reduz significativamente a rede mas se mostra limitada na sua aplicabilidade.

A segunda abordagem combina a técnica de *Geração dos Arcos Longos* com o problema de pseudo-designação. Esta técnica, proposta por Freling et al. (1995), é uma adaptação do princípio de Geração de Colunas da Programação Linear à teoria de Fluxo em Redes que, baseado nos custos relativos fornecidos pela solução corrente, informa se um arco deve ou não entrar no domínio do próximo problema. Esta adaptação, denominada Geração de Arcos, reduziu bastante o número de arcos a ser considerado no processo de otimização, além de permitir a inclusão de restrições operacionais ao modelo de fluxo em redes.

A duas metodologias foram testadas com dados reais das cidades de *Reading* no Reino Unido e de Sorocaba no Brasil. Os resultados obtidos foram comparados com aqueles provenientes do sistema heurístico *BOOST*, desenvolvido pelo grupo de programação de veículos e tripulação da Universidade de *Leeds*. Nestes testes foi determinado o melhor método de resolução, tendo em vista o desempenho computacional e a capacidade de incorporar características de problemas reais ao modelo matemático. Determinada uma metodologia de solução, foram realizados dois estudos de casos referentes às cidades de Santos e de Belo Horizonte, resultando na flexibilização e validação do método algoritmo.

ABSTRACT

This work explores network flow models to obtain an efficient representation for practical mass vehicle scheduling problems. These models are combined with different reduction techniques in order to minimise the network size for this sort of problems.

The first methodology combines the minimal cost flow model with the arcs reduction technique proposed by Bokinge and Hasselström (1980), which excludes certain kind of arcs from the network. These arcs are replaced by arcs already existing in the network. Although artificial arcs and nodes are included into the network, there is a considerably reduction in the problem dimension. It was used a version from this technique proposed by Freling et al. (1995), which was not very efficient in relation with side constraints inclusion.

A second approach applies an arc generation technique to the pseudo-assignment model for the vehicle scheduling problem. This technique used by Freling et al. (1995) is an adaptation from the column generation approach for linear programming to the network flow algorithms. Based on the reduced costs from the current solution, information are obtained for each arc to decide whether it must or not be included into the optimisation process. With the adaptation of the column generation to the network flow algorithms, it was possible to reduce considerably the total number of arcs, and also to consider some practical side constraints from public transport.

These two methodology were tested with Brazilian and English real cases. The results obtained were compared with those produced by the heuristic system BOOST, developed by the Vehicle and Crew Scheduling Group from the Leeds University. As a result from these tests, a method was chosen bearing in mind computational efficiency and the capability to represent practical features raising in real cases. Once selected the best method, was caried out case studies concerning to Santos and Belo Horizonte cities, which leaded to a flexibly and suitable algorithm.

Capítulo 1

Introdução

O Problema de Programar Veículos no Sistema de Transporte Público consiste em: *i*) determinar o número mínimo de veículos necessários para a operação, e *ii*) definir a seqüência das atividades executadas por cada veículo, no caso ônibus urbano de transporte coletivo, durante o dia. A definição da frota mínima está associada à minimização do custo capital e o seqüenciamento ótimo, que mostra desde a partida do ônibus da garagem no início da jornada, os detalhes das viagens realizadas, até o seu recolhimento final à garagem, está associado à minimização dos custos variáveis. Este trabalho aborda o Problema de Programação de Veículos do Sistema de Transporte Público com uma Única Garagem (*Single Depot Vehicle Scheduling Problem for Public Mass Transport*), utilizando Métodos de Fluxo em Redes para resolvê-lo.

1.1 Relevância do Tema

A privatização do sistema de transporte público, assim como o surgimento da concorrência advinda do crescimento do transporte alternativo, tem forçado as empresas de ônibus a utilizarem de maneira mais eficiente seus recursos materiais e humanos para que sejam lucrativas sem que haja um comprometimento na qualidade do serviço oferecido. Neste sentido todo o processo de planejamento das operações de ônibus deve ser revisado e otimizado sempre que possível.

O planejamento das operações é um trabalho complexo e por isso mesmo é dividido

em várias etapas, onde os dados de saída de uma etapa são utilizados como dados de entrada de etapas posteriores. As principais fases deste processo são:

- *a definição das rotas*: as rotas são definidas de acordo com uma pesquisa do tipo origem destino realizada junto à população e que mostra a direção do fluxo das pessoas dentro do plano municipal. As solicitações regionais para que necessidades locais sejam supridas também interferem na definição das rotas.

- *a definição das tabelas de horários*: a freqüência das viagens nas rotas é definida de acordo com estudos de previsão de demanda por viagens entre as várias regiões da cidade. A curva de demanda por viagens pode ter dois ou mais horários de pico: i) um pela manhã no sentido bairro-centro, ii) o pico do horário de almoço, e iii) um pico no final da tarde no sentido centro-bairro.

- *a definição das programações dos veículos*: a programação dos veículos depende das características das linhas, tais como: freqüência das viagens, distância entre os terminais e a distância entre os terminais e a garagem da empresa; e das características operacionais. Nesta etapa são definidos o número de veículos e a forma de operação da frota.

- *a definição dos turnos de trabalhos das tripulações*: para operar a frota são necessários motoristas e cobradores, ou seja, as tripulações. Nesta etapa é realizada a alocação das tripulações aos respectivos veículos, de tal maneira que os turnos de trabalho respeitem as normas trabalhistas tais como: tempo máximo de trabalho, tempo mínimo de descanso entre os turnos, pausa para alimentação, etc. A combinação destes fatores gera as oportunidades de troca da tripulação, que irão determinar os turnos das tripulações.

Na prática operacional brasileira a tripulação "acompanha"o ônibus, que por sua vez é cativo à linha. Em outros países como no Reuno Unido e na Alemanha, tanto o veículo quanto a tripulação podem atuar em mais de uma linha em um mesmo turno de trabalho. Operações mais flexíveis, que permitem que veículos e tripulações troquem de linha durante a jornada de trabalho, conta com um número maior de alternativas de programação.

Dentre as etapas do planejamento das operações, a definição das rotas e a definição das tabelas de horários normalmente são feitas pelos órgãos de gerência do transporte público, restando aos operadores definir a programação dos veículos e os turnos de trabalho das tripulações. Logo é nestas duas etapas do processo que as empresas de transporte público podem aprimorar suas atividades tornando-se cada vez mais eficientes.

A importância deste estudo vai além da resolução eficiente do problema, visto que diversos algoritmos de programação de veículos com múltiplas garagens e com vários tipos de ônibus tem como ponto de partida o problema de programação de veículos com uma única garagem (Daduna e Paixão 1995). Além disso, os principais métodos de programação da tripulação partem da programação dos veículos para então gerar a programação da tripulação (Carraresi e Gallo 1984, Desrochers e Soumis 1989, Fores 1996).

Outro fator que se destaca é o relacionamento da etapa de programação dos veículos com suas etapas vizinhas. A programação de veículos está intimamente ligada à definição da tabela de horários, da mesma forma que ela interage com a programação da tripulação.

Portanto, desenvolver uma metodologia que resolva eficientemente o problema de programação de veículos com uma única garagem é o ponto de partida para a abordagem de outros problemas, tais como a definição de tabelas de horários, a geração de programações de veículos mais complexas, e a programação da tripulação.

1.2 Objetivo do Trabalho

Este trabalho tem como finalidade desenvolver uma metodologia para resolver eficientemente o Problema de Programação de Veículos com uma Única Garagem (PPV). As principais dificuldades na resolução desse problema se deve basicamente a dois motivos: *i*) a existência de um grande número de variáveis, e *ii*) a dificuldade de considerar todas as restrições do problema real. Apesar destas dificuldades, vários métodos de resolução são apresentados na literatura e estudos de casos demostram que há um ganho significativo quando métodos de otimização são colocados em

prática (Wren e Kwan 1999, Smith e Wren 1981, Gavish et al. 1978, Wren 1972).

Uma vez definido um método eficiente de resolução do problema, estudos de caso são realizados visando validar e flexibilizar o modelo proposto.

1.3 Metodologia Empregada

São inúmeros os trabalhos que utilizam os modelos de fluxo em redes para resolver o problema de programação de veículos, o que pode ser verificado nos trabalhos de revisão bibliográfica de Carraresi e Gallo (1984) e de Daduna e Paixão (1995). Isto se deve pela eficiência de tais modelos em representar o problema e pela simplicidade de implementação dos algoritmos de fluxo em redes, os quais em sua maioria apresentam complexidade polinomial. Além disso, uma vez consideradas apenas unidades inteiras de fluxo, a solução ótima será necessariamente inteira, evitando problemas com erros de arredondamento e de instabilidade numérica. Por estes motivos foram escolhidos os algoritmos de fluxo em redes para abordar o problema.

Neste trabalho são exploradas duas técnicas de redução da rede que representa o problema (Freling et al. 1995). Os modelos foram adaptados para que o problema de fluxo em redes pudesse ser resolvido com o algoritmo *Out-of-Kilter* (Fulkerson 1961, Dunlay e Gualda 1979). Foi escolhido o algoritmo de fluxo em redes *Out-of-Kilter* devido à existência de uma implementação básica no Laboratório de Planejamento de Transportes (LPT) do Departamento de Engenharia de Transporte. Assim, são derivadas duas metodologias para resolver o problema.

Na primeira metodologia o problema é formulado como um problema de fluxo com custo mínimo, aplicando-se então a técnica de redução da rede proposta por Bokinge e Hasselström (1980). A idéia é descartar os arcos que ligam viagens com um longo tempo de espera entre elas. Estes arcos, ditos arcos longos, são substituídos pelos arcos já existentes na rede que ligam a viagem à garagem. Nós adicionais são incluídos à rede para representar a garagem no decorrer do intervalo de programação. Tais nós possibilitam o retorno temporário dos veículos à garagem durante a operação.

A segunda metodologia explorada neste trabalho formula o problema como um pro-

blema de pseudo-designação e utiliza a técnica de geração de arcos para resolvê-lo de maneira eficiente. A geração de arcos é uma adaptação da técnica de geração de colunas à teoria de fluxo em redes e consiste em incluir novos arcos à rede tendo como base seus custos relativos. A partir de um sub-conjunto inicial de arcos, resolve-se o problema e novos arcos são incluídos à rede caso seus custos relativos indiquem uma possível melhoria na solução. O problema é resolvido novamente sempre que algum arco tiver sido incluído à rede. Quando não houver inclusão de arcos, a solução corrente será ótima. Esta técnica é mais geral e permite a inclusão de varias restrições operacionais que aparecem com freqüência no transporte público. Paralelamente a esta pesquisa Löbel (1999) utiliza a geração de arcos para resolver o Problema de Programação de Veículos com Várias Garagens.

Estas duas metodologias foram testadas com dados reais das cidades de *Reading* (Reino Unido), Sorocaba e seus resultados são comparados com os resolvidos fornecidos pelo sistema heurístico *BOOST* concebido por Wren (1972) e na versão atual implementada por Kwan e Rahin (1999). Nesta fase são comparados os resultados obtidas através dos modelos propostos com os resultados produzidos por um sistema heurístico de referência internacional na área de programação de veículos.

Uma vez desenvolvida e testada a metodologia para resolver o problema de programação de veículos com uma única garagem, foi verificada a potencialidade do método na abordagem de caso reais brasileiros pertinentes às cidades de Santos e Belo Horizonte. Neste etapa do trabalho foram introduzidas ao modelo restrições adicionais peculiares à realidade operacional.

Embora a resolução do problema proposto abra um caminho para resolver outros problemas mais complexos, este trabalho tem seu foco no problema básico de programação de veículos. Sendo assim, foram realizados estudos de casos que permitiram observar e incluir características particulares de diferentes realidades brasileiras ao modelo, tornando-o mais flexível e adequado à prática.

1.4 Estrutura do Trabalho

O próximo capítulo apresenta os conceitos básicos e a notação adotada. Nos capítulos 3 e 4 é feito um levantamento bibliográfico dos principais modelos e métodos de resolução do problema. No capítulo 5 são apresentados os testes de eficiência dos métodos de redução da rede. O capítulo 6 apresenta a método proposto, suas adaptações aos casos estudados, assim como os detalhes da sua implementação. No capítulo 7 são expostos os casos estudados, seguido das conclusões na capítulo 8.

Capítulo 2

Conceitos e Notação em Redes

2.1 Introdução

Neste capítulo será apresentada a notação básica utilizada no texto, assim como conceitos de programação linear, de teoria dos grafos e de fluxo em redes indispensáveis para o entendimento das técnicas abordadas neste trabalho. Os leitores habituados a tais conceitos podem seguir a leitura para o capítulo seguinte. A notação adotada assim como a descrição de métodos matemáticos empregados neste trabalho está baseada nas seguintes referências: Ahuja et. al. (1993) e Murty (1990).

2.2 Fluxo em Redes

Uma **rede** é um par de conjuntos $G = (N,\ A)$, onde N é um conjunto de *nós* (também dito *pontos* ou *vértices*) e A é um conjunto *linhas*, cada linha ligando pares de nós distintos. Uma linha que liga os nós i e j é chamada de *arco* se ela só puder ser usada em uma determinada direção, seja de i para j. Neste caso o arco será denotado pelo *par ordenado* (i, j). O arco (i, j) *sai* do nó i e *chega* no nó j, portanto i é a *cauda* e j é a *cabeça* de (i, j). Uma rede G é dita *rede direcionada* se o seu conjunto A for constituido apenas por arcos. Pode-se ainda associar um custo por unidade de fluxo que atravessa cada arco, c_{ij}. Considerando a rede direcionada $G = (N,\ A,\ c)$, são apresentadas as seguintes definições:

Grau: O *grau interno* de um nó em uma rede é o número de arcos que chegam no nó e o seu *grau externo* é o número de arcos que saem do nó. O *grau* de um nó é a soma do seu grau interno com o seu grau externo.

Lista de adjacência: A *lista dos arcos adjacentes* $A(i)$ de um nó corresponde ao conjunto de arcos que saem dele, ou seja, $A(i) = \{(i,j) \in A : j \in N\}$. Assume-se que a lista de adjacência $A(i)$ está disposta em ordem crescente em relação à cabeça dos arcos contidos nela.

Caminhos e ciclos: Um *caminho* $P = (n_0, a_1, n_1, a_2, ..., a_k, n_k)$ de n_0 para n_k é uma seqüência de n nós distintos $n_0, n_1, ..., n_k$ e m arcos $a_1, a_2, ..., a_k$, onde a_k é o arco (n_{k-1}, n_k) ou o arco (n_k, n_{k-1}). Se $a_k = (n_{k-1}, n_k)$ este arco está no sentido do caminho que *conecta* o nó n_0 ao nó n_k, e portanto ele é dito *arco direto* no caminho; caso contrário, se $a_k = (n_k, n_{k-1})$, ele estará no sentido oposto ao do caminho, então ele será dito *arco reverso* no caminho. Um *caminho direcionado* de n_0 para n_k é um caminho composto apenas por arcos diretos.

O *comprimento* de um caminho é igual ao número de arcos que o constitui. Um *caminho mínimo* de n_0 para n_k é um caminho de menor comprimento entre n_0 e n_k.

Um caminho de n_0 para n_0 contendo pelo menos dois arcos é um *ciclo*.

Conectividade: Uma rede é dita *conectada* (ou *conexa*) se existe pelo menos um caminho ligando qualquer par de nós na rede, caso contrário ela é dita *desconectada* (ou *conectada*). Uma rede é dita *fortemente conectada* se existe pelo menos um caminho direcionado ligando qualquer par de nós da rede.

Corte: Um corte é um particionamento do conjunto de nós N em dois subconjuntos, $(S, \bar{S})$, onde $\bar{S} = N - S$. Todo corte define um conjunto de arcos com um extremo em S e outro em $\bar{S}$. Este conjunto de arcos é denotado por $[\mathbf{X}, \bar{\mathbf{X}}]$.

Rede bipartida: A rede G será bipartida se seu nós puderem ser particionados em dois subconjuntos N_1 e N_2 tal que qualquer arco $(i,j) \in A$ terá $i \in N_1$ e $j \in N_2$.

Capacidade e limite inferior: A *capacidade* u_{ij} e o *limite inferior* l_{ij} de um arco (i, j), são respectivamente as quantidade máxima e mínima de fluxo que pode ser transportada de i para j através deste arco. Naturalmente que $l_{ij} \leq u_{ij}$.

Rede capacitada: Uma *rede capacitada* $G = (N, A, l, u, c)$ é aquela cujos arcos são providos de capacidade e limitante inferior.

Oferta e demanda nos nós: A cada nó $i \in N$ associa-se um número inteiro $b(i)$ que representa a oferta ou demanda por unidades de fluxo no nó. Se $b(i) > 0$, i é um *nó de oferta*; se $b(i) < 0$, i é um *nó de demanda* de fluxo; e se $b(i) = 0$, i é um *nó de transbordo*.

Origem, destino: Na maioria dos problemas de fluxo, a rede pode ser construída contendo um nó de oferta ou a *origem* de onde emana todo o material a ser transportado pela rede, assim como um nó de demanda ou o *destino* para onde todo o material deve escoar. Todos os outros nós são de transbordo.

Fluxo factível: Um fluxo $f = (f_{ij})$ é um vetor que especifica a quantidade de fluxo em cada arco da rede. Para problemas de fluxo em redes capacitadas, um *fluxo factível* satisfaz as seguintes restrições:

$$l_{ij} \leq f_{ij} \leq u_{ij} \quad \forall (i, j) \in A \tag{2.1}$$

e

$$\sum_{j \in A_i} f_{ij} - \sum_{j \in B_i} f_{ji} = \begin{cases} v & \text{se } i = s \\ -v & \text{se } i = t \\ 0 & \text{caso contrário} \end{cases} \tag{2.2}$$

onde $A_i = \{j : (i, j) \in A\}$ e $B_i = \{j : (j, i) \in A\}$, v é a quantidade de fluxo que sai da origem e chega na destino, conhecida como *valor do fluxo f*. As restrições (2.2) são conhecidas como restrições de *conservação do fluxo* e garantem que o total de fluxo que chega é igual à quantidade de fluxo que sai dos nós intermediários.

Caminho de aumento de fluxo: Seja f um fluxo factível com valor v em uma rede direcionada e capacitada G. Um caminho P com origem s e destino t é um *caminho de aumento de fluxo em relação a f* se:

$$\begin{array}{ll} f_{ij} < u_{ij} & \text{onde } (i, j) \text{ é arco direto em } P \\ f_{ij} > l_{ij} & \text{onde } (i, j) \text{ é arco reverso em } P \end{array}$$

Assim, seja

$$\begin{aligned} \varepsilon_1 &= \textit{minimo}\,\{u_{ij} - f_{ij} : (i,j) \text{ é arco direto em } P\} \\ \varepsilon_2 &= \textit{minimo}\,\{f_{ij} - l_{ij} : (i,j) \text{ é arco reverso em } P\} \\ \varepsilon &= \textit{minimo}\,\{\varepsilon_1, \varepsilon_2\} \end{aligned}$$

Logo, define-se o novo fluxo $\hat{f}$ por:

$$\begin{aligned} \hat{f}_{ij} &= f_{ij} && \text{se } (i,j) \notin P \\ &= f_{ij} + \varepsilon && \text{se } (i,j) \text{ é arco direto em } P \\ &= f_{ij} - \varepsilon && \text{se } (i,j) \text{ é arco reverso em } P \end{aligned}$$

Então, $\hat{f}$ é um fluxo factível com valor $\hat{v} = v + \varepsilon$, e ε é a *capacidade de aumento no caminho P*.

2.3 Complexidade, Problemas *P* e *NP-Completos*

Os problemas de programação são de natureza complexa, necessitando técnicas especializadas de solução. Considerando um determinado algoritmo de solução para um dado problema, é importante ter uma previsão do esforço computacional necessário para resolver o problema de um certo tamanho. Assim, a escolha do algoritmo ou abordagem de solução pode ser feita em função do tempo de processamento.

Vários algoritmos podem resolver um determinado problema em tempo polinomial, isto é, o número de operações matemáticas $(+, -, \times, \div)$ executadas, no pior caso, pode ser limitado por uma função polinomial dos parâmetros do problema. Neste caso o algoritmo tem *complexidade polinomial*, caso contrário, o algoritmo tem *complexidade exponencial*. Vale ressaltar que é o algoritmo que tem complexidade polinomial e não o problema em si. Uma característica importante dos problemas que podem ser resolvidos com um algoritmo polinomial é que torna-se previsível como o tempo de processamento cresce em relação ao tamanho do problema.

Considerando a classificação dos algoritmos em polinomial e exponencial, pode-se determinar duas classes de problemas.

P = todos os problemas para os quais se conhece um algoritmo polinomial.
Exemplo: problema do caminho mínimo, problema do fluxo máximo, problema de fluxo com custo mínimo, problemas de designação e emparelhamento.

NP = todos os problemas para os quais se conhece um algoritmo exponencial. Exemplo: problema do caixeiro viajante, problema do caminho Hamiltoniano, problema do caminho mínimo com restrições, problema de fluxo com variáveis inteiras.

Um problema é *redutível polinomialmente* em outra, se for possível transformar um problema no outro através de um número polinomial de operações. Considere um problema *NP* tal que todos os outros problemas *NP* sejam polinomialmente redutíveis a ele, então o problema é considerado *NP-completo*. Apesar de improvável, existe um grande número de problemas *NP-completo*, incluindo a maioria dos problemas de programação. Cook (1971) foi o primeiro a estudar problemas P e *NP* introduzindo o conceito de *NP-Completeza* (*NP-Completeness*). Ele mostrou que todo problema em *NP* pode ser polinomialmente reduzido a um problema arquétipo de reconhecimento chamado de *satisfiability*, que é *NP-completo*.

French (1982) faz a seguinte distinção entre os problemas:

Problema de Otimização: quando se pretende encontrar uma solução ótima. Exemplo: Minimizar $F(x)$ sujeito a um comnjunto de restrições C.

Problema de Reconhecimento: quando se procura detectar se uma solução com determinada característica existe. A característica pode estar associada ao custo. Exemplo: Dado o conjunto de restrições C, existe uma solução $\bar{x}$ tal que $F(\bar{x}) \leq K$?. O problema de reconhecimento tem como resposta "Sim" ou "Não".

Se existe um valor particular K para o qual o problema de reconhecimento é *NP-completo*, então ele é *NP-completo* para todos os valores de K, e o correspondente problema de otimização é um problema *NP-difícil*. O termo *NP-completo* se refere ao problema de reconhecimento, enquanto o termo *NP-difícil* é atribuído ao problema de otimização. Como existe uma equivalência entre os dois problemas, os termos são utilizados como sinônimos de problemas para os quais não existe um algoritmo de solução com complexidade polinomial.

2.4 O Problema de Fluxo com Custo Mínimo

Seja $G = (N, A, l, u, c, b)$, onde N é o conjunto dos nós, A o conjunto dos arcos, l, u e c os limitantes inferiores, superiores e respectivos custos nos arcos, e b a oferta/demanda nos nós. Então o problema de encontrar de o fluxo que atenda à oferta/demanda nos nós com menor custo é dado por:

$$\text{Min} \sum_{(i,j)\in A} c_{ij} f_{ij} \qquad \text{sujeito a} \tag{2.3}$$

$$\sum_{j:(i,j)\in A} f_{ij} - \sum_{j:(j,i)\in A} f_{ji} = b(i) \tag{2.4}$$

$$l_{ij} \leq f_{ij} \leq u_{ij} \qquad \forall\ (i,j) \in A \tag{2.5}$$

A expressão (2.3) visa minimizar o custo total do fluxo na rede, enquanto as restrições (2.4)e (2.5) garantem a factibilidade do fluxo resultante. No problema dual, existem variáveis duais associadas às equações de conservação de fluxo associada a cada nó da rede (2.4). Seja π_i a variável associada ao nó $i \in A$. Estas variáveis são chamadas de *variáveis dos nós*, *preço nos nós*, ou *potencial nos nós*. A condição das folgas complementares para a otimalidade de um fluxo f é a existência do vetor dual π satisfazendo:

$$\begin{array}{llll} \text{se} & \pi_j - \pi_i > c_{ij} & \text{então} & f_{ij} = u_{ij} \\ \text{se} & \pi_j - \pi_i < c_{ij} & \text{então} & f_{ij} = l_{ij} \\ \text{se} & \pi_j - \pi_i = c_{ij} & \text{então} & l_{ij} \leq f_{ij} \leq u_{ij} \end{array} \tag{2.6}$$

O algoritmo *out-of-kilter* trabalha com um *par vetores solução* $(f = (f_{ij}), \pi)$ primal e dual factível, e muda alternadamente uma solução de cada vez, de tal forma que as condições das folgas complementares sejam satisfeitas.

2.4.0.1 Problema de Circulação em Redes Capacitadas

O problema de circulação é um problema de fluxo com custo mínimo que contém somente nós de transbordo, isto é, $b(i) = 0, \forall i \in N$. Neste problema procura-se um fluxo ($f = (f_{ij})$ que esteja dentro dos limitantes inferiores e superiores l_{ij} e u_{ij} impostos ao arco (i, j), e cujo custo total seja mínimo.

2.5 O Problema de Designação e Pseudo-Designação

Dada uma matriz de custos $C = (c_{ij})$, define-se o *problema de designação* da seguinte maneira:

$$\text{Min} \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \quad \text{sujeito a} \tag{2.7}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \text{para } i = 1, .., n \tag{2.8}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad \text{para } j = 1, .., n \tag{2.9}$$

$$x_{ij} \geq 0 \text{ para todo } i, j = 1, .., n \tag{2.10}$$

Se $x = (x_{ij})$ for uma solução básica do problema acima, então $x_{ij} = 0$ ou $x_{ij} = 1$, qualquer que seja $i, j \in \{1, ..n\}$. Tal matriz é denominada uma *designação* ou uma *matriz de permutação*. Em uma designação $x = (x_{ij})$, a posição (i, j) está designada se $x_{ij} = 1$, e neste caso a linha i está alocada ou emparelhada com a coluna j. Além disso, exite uma única designação para cada linha e para cada coluna. Logo, define-se uma designação $x = (x_{ij})$ pelo conjunto $\{(i, j) : x_{ij} = 1\}$, ou seja, o conjunto de posições com alocação.

Se cada linha e cada coluna for representada por um nó, e as diferentes posições forem substituidas por arcos, tem-se uma *rede bipartida* $G = (N_1, N_2, A)$ com $|N_1| = |N_2|$ e se $(i, j) \in A$ então $i \in N_1$ e $j \in N_2$. O custo c_{ij} pode ser considerado como o custo no arco que liga a linha i à coluna j e o problema de designação escrito como um problema de fluxo em redes fica assim:

$$\text{Min} \sum_{(i,j)\in A} c_{ij}x_{ij} \quad \text{sujeito a} \tag{2.11}$$

$$\sum_{j:(i,j)\in A} x_{ij} = 1, \quad \forall\ i \in N_1 \tag{2.12}$$

$$\sum_{i:(i,j)\in A} x_{ij} = 1, \quad \forall\ j \in N_2 \tag{2.13}$$

$$x_{ij} \geq 0 \text{ para todo } (i, j) \in A \tag{2.14}$$

Dada uma rede, um *emparelhamento* é uma subrede na qual cada nó tem grau menor ou igual a 1. Um *emparelhamento perfeito* ou *emparelhamento completo* é uma subrede na qual cada nó tem grau igual a 1. Tomando um emparelhamento

completo numa rede bipartida; definindo $x_{ij} = 1$ se a linha i estiver ligada à coluna j, e $x_{ij} = 0$ caso contrário, resulta na matriz $x = (x_{ij})$ que corresponde a uma designação. Portanto, o problema colocado em (2.7) - (2.10) equivale a resolver o problema do emparelhamento completo com custo mínimo. Esta é a razão pela qual o problema de designação também é conhecido com o *problema do emparelhamento bipartido*.

Uma *Pseudo-designação* é uma designação na qual as restrições (2.12) e (2.13) são válidas para todos os nós, com exceção de um em cada conjunto. Ou seja, um nó do conjunto N_1 está associado a mais de um nó do conjunto N_2 e vice-versa. Considerando que os nós $s \in N_1$ e $t \in N_2$ violam as restrições de igualdade da designação, para um certo $n > 1$ o problema de pseudo-designação fica com a seguinte formulação matemática:

$$\text{Min} \sum_{(i,j)\in A} c_{ij}x_{ij} \qquad \text{sujeito a} \tag{2.15}$$

$$\sum_{j:(i,j)\in A} x_{ij} = 1, \qquad \forall\ i \in N_1 - \{s\} \tag{2.16}$$

$$\sum_{i:(i,j)\in A} x_{ij} = 1, \qquad \forall\ j \in N_2 - \{t\} \tag{2.17}$$

$$\sum_{j:(i,j)\in A} x_{sj} = n \tag{2.18}$$

$$\sum_{i:(i,j)\in A} x_{it} = n \tag{2.19}$$

$$x_{ij} \geq 0 \qquad \text{para todo } (i,j) \in A \tag{2.20}$$

2.6 O Algoritmo *Out-of-Kilter*

Considere o problema de fluxo com custo mínimo (2.3) - (2.5) na rede direcionada $G = (N, A, l, u, c)$ com $|N| = n$ e $|A| = m$. Para resolver o problema, o método *out-of-kilter* , proposto por Fulkerson (1961), pode ser inicializado com qualquer par de vetores de fluxo e de preço nos nós (f, π). Entretanto, a eficiência do algoritmo aumenta consideravelmente se o fluxo inicial for factível. O método se alterna entre uma rotina de mudança no fluxo, quando o vetor de preços nos nós permanece inalterado, e uma rotina de mudança no vetor de proço nos nós, quando o vetor de fluxo permanece fixo. Desta forma, a cada iteração o par de vetores (f, π) se

aproxima da solução ótima do problema.

O *status kilter* de um arco (i, j) é determinado pelas condições apresentadas na tabela (2.1).

Tabela 2.1: Diferentes tipos de arcos com suas respectivas características.

Tipo do arco	Condições	*Status*
α	$\pi_j - \pi_i < c_{ij}$ e $f_{ij} = l_{ij}$	*in-kilter*
β	$\pi_j - \pi_i = c_{ij}$ e $l_{ij} \leq f_{ij} \leq u_{ij}$	*in-kilter*
γ	$\pi_j - \pi_i > c_{ij}$, e $f_{ij} = u_{ij}$	*in-kilter*
a	$\pi_j - \pi_i < c_{ij}$ e $f_{ij} > l_{ij}$	*out-of-kilter*
b	$\pi_j - \pi_i > c_{ij}$, e $f_{ij} < u_{ij}$	*out-of-kilter*

Os pares (f, π) que satisfazem as condições de otimalidade são do tipo α, β, γ e portanto são ditos arcos *in-kilter*. Aqueles que violam as condições de otimalidade correspondem aos arcos dos tipos a, e b, e são denominados arcos *out-of-kilter* .

Dado o par de vetores (f, π), um arco β pode ainda ser classificado de três formas distintas, dependendo do seu fluxo.

Classe	Condição
$\beta - superior$	$f_{ij} = u_{ij}$
$\beta - interior$	$l_{ij} < f_{ij} < u_{ij}$
$\beta - inferior$	$f_{ij} = l_{ij}$

Para assegurar que arcos *in-kilter* permaneçam sempre *in-kilter* e que arcos *out-of-kilter* sempre melhorem seu *status de kilter*, são permitidas apenas determinadas mudanças no fluxo e nos preços dos nós. As mudanças permitidas são:

Em cada iteração o algoritmo seleciona um arco *out-of-kilter* e tenta transformá-lo em um arco *in-kilter*. Suponha que o arco (p, q) tenha sido selecionado, então o algoritmo tenta passá-lo para *in-kilter* mudando seu fluxo. Se (p, q) for um arco tipo a, o fluxo f_{pq} deve decrescer. Para manter a factibilidade, a quantidade decrescida em f_{pq} de ser enviado de p para q por um caminho diferente, i.e, um caminho que não contenha o arco (p, q). Esta operação de mudança de fluxo é dita *redirecionamente de fluxo*, onde p é a origem e q é o destino do redirecionamento. Se o arco selecionado

Tabela 2.2: Diferentes tipos de arcos com suas respectivas características.

Tipo do arco	Mudanças permitidas
a	f_{ij} pode diminuir até l_{ij} $\pi_j - \pi_i$ pode aumentar até c_{ij}
b	f_{ij} pode aumentar até u_{ij} $\pi_j - \pi_i$ pode diminuir até c_{ij}
β	f_{ij} pode variar livremente entre l_{ij} e u_{ij} $\pi_j - \pi_i$ não pode mudar para arcos $\beta - interiores$ $\pi_j - \pi_i$ pode diminuir para arcos $\beta - inferiores$ $\pi_j - \pi_i$ pode aumentar para arcos $\beta - superiores$
α	f_{ij} não pode mudar $\pi_j - \pi_i$ pode variar em $(-\infty, c_{ij}]$
γ	f_{ij} não pode mudar $\pi_j - \pi_i$ pode variar em $[c_{ij}, +\infty)$

for do tipo b, o fluxo f_{pq} deve ser aumentado e neste caso a origem e o destino do redirecionamento são q e p respectivamente.

Dado o par de soluções primal-dual $(f = (f_{ij}), \pi)$, seja o arco (p, q) selecionado e $\bar{s}$ e $\bar{t}$ origem e destino do redirecionamento. Define-se $\nu = f_{pq} - l_{pq}$ se (p, q) for do tipo a. Para colocar (p, q) *in-kilter* apenas com a mudança do fluxo, é necessário redirecionar ν unidades de fluxo de $\bar{s}$ para $\bar{t}$. Para tanto é necessário encontrar um caminho de aumento do fluxo (CAF) de $\bar{s}$ para $\bar{t}$ em G, que permita as mudanças de fluxo desejadas, isto é, todo arco direto no caminho deve ser do tipo b ou β. Analogamente, define-se $\nu = u_{pq} - f_{pq}$ se (p, q) for do tipo b. Neste caso, é necessário encontrar um CAF de $\bar{s}$para $\bar{t}$no qual todos arcos reversos no caminho sejam do tipo a ou β. O algoritmo tenta encontrar o menor CAF de $\bar{s}$ para $\bar{t}$ usando uma rotina de crescimento de árvore, cujos arcos permitam uma mudança desejável de fluxo.

Seja ϵ_1 a capacidade de um dado CAF P, então $\epsilon_0 = \min\{\nu,\ \epsilon_1\}$. Assim o redirecionamento de fluxo usado P consiste em adicionar ϵ_0 ao fluxo nos arcos diretos de P e no fluxo de (p, q) se este for do tipo b, e subtrair ϵ_0 nos arcos reversos de P e de (p, q) se este for do tipo a.

Se a rotina de crescimento da árvore não for capaz de encontrar um CAF, ela terminá com um conjunto de nós rotulados $\mathbf{X}$ contendo $\bar{s}$ e que não contém $\bar{t}$. Isso implica que nesta iteração não é possível melhorar o *status kilter* do arco selecionado através

da mudança do seu fluxo. Resta então tentar esta melhoria pela mudança no preço dos nós. Seja $\bar{\mathbf{X}}$ o complemento se $\mathbf{X}$, então os arcos diretos do corte $[\mathbf{X}, \bar{\mathbf{X}}]$, neste iteração, podem ser do tipo $a,\ \alpha,\ \gamma$ ou $\beta - superior$, mas nunca do tipo a.

Seja:

$$A^1 = \{(i,j) : (i,j) \text{ é arco direto do tipo } a \text{ ou } \alpha \text{ em } (\mathbf{X}, \bar{\mathbf{X}})\} \tag{2.21}$$

$$A^2 = \{(i,j) : (i,j) \text{ é arco reverso do tipo } b \text{ ou } \gamma \text{ em } (\bar{\mathbf{X}}, \mathbf{X})\} \tag{2.22}$$

$$\delta = \min \{|\pi_j - \pi_i - c_{ij}| : (i,j) \in A^1 \cup A^2\} \tag{2.23}$$

então, em cada arco de A^1 (A^2) é possível aumentar (diminuir) sua tensão pela mudança nos preços nos nós. Assim, δ é simultaneamente o maior aumento possível na tensão nos arcos em A^1, e a maior diminuição possível da tensão nos arcos em A^2. O novo vetor de preços terá a forma $\hat{\pi} = (\hat{\pi}_i)$ onde $\hat{\pi}_i = \pi_i$ se $i \in \mathbf{X}$, ou $\hat{\pi}_i = \pi_i + \theta$ se $i \in \bar{\mathbf{X}}$, para um valor θ escolhido adequadamente. Consequentemente, a tensão em um arco não muda se ele estiver completamente contido em X ou em $\bar{\text{X}}$. Por outro lado, ela aumenta de θ unidades se estiver em (X, $\bar{\text{X}}$) e diminui a mesma quantidade θ se estiver em ($\bar{\text{X}}$, X). Para que estas mudanças estejam de acordo com o previsto na tabela (2.2), $\theta \leq \delta$. O valor de θ que faz com que o arco chegue o mais próximo possível do *status in-kilter* é $\theta = \delta$. Desta forma, no novo par $(f, \bar{\pi})$, o *status kilter* de cada arco em G é igual ou melhor ao *status* anterior. Todos os arcos que estavam *in-kilter* anteriormente permanecerão *in-kilter* após cada iteração. O algoritmo pode ser sintetizado pelos cinco passos descritos abaixo.

Algoritmo Out-of-Kilter

Passo 1 - Inicialização

Seja (f^0, π^0) um par de vetores com fluxo factível. Se este satisfaz as condições de otimalidade na tabela (2.1), ele é ótimo: terminar. Caso contrário vá para o passo 2.

Passo 2 - Seleção de um arco

Selecionar um arco (p,q) *out-of-kilter* e vá para o passo 3.

Passo 3 - Rotina de rotulamento

Identifique a origem do CAF $\bar{s}$ e rotule-a com 0. Faça $Lista = \{\ \bar{s}\}$. Vá para o subpasso 1.

Subpasso 1 - Seleção de um nó da lista a ser pesquisado

Se a $Lista = \emptyset$, vá para o passo 5. Caso contrário escolha o primeiro nó para ser pesquisado, retire-o da lista e vá para o subpasso 2.

Subpasso 2 - Pesquisa do nó selecionado

Seja i o nó a ser pesquisado, e $(f = (f_{ij}), \pi)$ o par corrente de vetores solução.

(i) Rotulamento direto

Rotular cada nó j ainda não rotulado nesta iteração, tal que (i, j) é um arco tipo b ou β e $f_{ij} < u_{ij}$ com $(i, +)$, e incluí-lo no começo da lista.

(ii) Rotulamento reverso

Rotular cada nó j ainda não rotulado, tal que (j, i) é um arco tipo a ou β e $f_{ij} > l_{ij}$ com $(i, -)$, e incluí-lo no começo da lista.

Se o destino do reroteamento $\bar{t}$ tiver sido rotulado, existe um ciclo negativo (*breakthrough*), e foi encontrado um CAF de $\bar{s}$ para $\bar{t}$. Vá para o passo 4; caso contrário volte ao subpasso 1.

Passo 4 - Reroteamento do fluxo

Encontre um CAF P, de $\bar{s}$ para $\bar{t}$ percorrendo o rotulamento inversamente inicando em $\bar{t}$. Quando o caminho P é combinado com o arco selecionado (p, q), forma-se um ciclo com custo residual negativo. Cancele o ciclo negativo em f, gerando o novo vetor de fluxo $\hat{f}$. Calcule o novo *status kilter* de cada arco de G, se todos os arcos estiverem *in-kilter* o par de vetores corrente é ótimo, termine o algoritmo. Se o arco escolhido ainda estiver *out-of-kilter*, apague todo o rotulamento e volte ao passo 3 considerando o mesmo arco (p, q). Se algum outro arco estiver *out-of-kilter* volte ao passo 2.

Passo 5 - Mudança do preço nos nós

Seja $\mathbf{X}$ o conjunto de nós rotulados nesta iteração, e $\bar{\mathbf{X}}$ o seu complementar.

Calcule o valor de δ de acordo com a expressão (2.23). Faça o novo vetor de preços nos nós igual a $\hat{\pi} = (\hat{\pi}_i)$ onde $\hat{\pi}_i = \pi_i$ para todo $i \in \mathbf{X}$, e $\hat{\pi}_i = \pi_i + \delta$ para todo $i \in \bar{\mathbf{X}}$. Encontre o novo *status kilter* de cada arco no corte $[\mathbf{X}, \bar{\mathbf{X}}]$ com respeito ao novo par $(f, \hat{\pi})$. Se todos os arcos estiverem agora *in-kilter* o par de vetores é ótimo, termine o algoritmo. Se o arco escolhido ainda estiver *out-of-kilter*, faça $Lista = \mathbf{X}$, e recomece a geração da árvore voltando ao subpasso 1 do passo 3. Caso contrário, volte ao passo 2.

2.7 Conclusão

Neste capítulo foi definida a notação e apresentados os conceitos básicos de fluxo em redes, teoria dos grafos, complexidade algoritmica e de problemas P e NP. Embora estes conceitos estejam difundidos entre aqueles que atuam na área, eles servem de base para que o leitor iniciante possa acompanhar o desenvolvimento deste trabalho.

Capítulo 3

O Problema na Literatura

3.1 Introdução

Este capítulo apresenta o problema de programação de veículos, assim como sua contextualização em relação às diferentes formas de operação encontradas na prática. O problema é definido e diferentes modelos matemáticos são apresentados, seguidos dos seus respectivos métodos de resolução. Uma breve descrição do sistema heurístico utilizado neste trabalho, o *BOOST*, pode ser encontrada no final deste capítulo.

3.2 O Problema de Programação de Veículos

O Problema de Programação de Veículos foi um dos primeiros problemas de transporte público a ser estudado e resolvido com o auxílio de computadores. Os primeiros trabalhos surgiram entre o final dos anos 60 e o início dos anos 70, quando foram utilizados tanto métodos exatos (Kirkman 1968) quanto métodos heurísticos (Saha 1970, Wren 1972).

Este problema tem como dados iniciais as viagens descritas por uma tabela de horários, às quais devem ser atribuídos veículos, tal que:

- cada viagem seja executada uma única vez,
- existe um número limitado para cada tipo de veículo.

- o número de veículos seja mínimo, e
- o custo operacional seja minimizado.

A situação mais simples, mas nem por isso menos importante, trata de uma única garagem, e um único tipo de veículos. Este é o *problema básico de programação de veículos* descrito abaixo.

O Problema Básico

Este problema tem grande importância não só individualmente, como também pela sua relação direta com problemas mais complexos que surgem na operação do sistema de transporte público. Ao resolver o problema básico, são resolvidos também inúmeros casos práticos, resultando na programação ótima da frota. Em alguns casos são necessárias algumas adaptações ao modelo para que este considere restrições específicas da situação estudada.

Problemas mais complexos surgem principalmente pela: existência de mais de uma garagem, existência de diferentes tipos de veículo em operação e limitação do tempo de operação. As resolução clássicas destes problemas passam pela resolução do problema com uma única garagem e um único tipo de veículo; portanto, ele deve ser resolvido com a maior eficiência possível. Uma vez que este problema é uma peça fundamental na resolução de outros problemas mais complicados, ele é denominado Problema Básico de Programação de Veículos (Daduna e Paixão 1995).

Devido à existência de diferentes formas de operacão, uma série de características distintas deve ser considerada, visto que elas determinam o tipo e a complexidade do problema. A seguir são destacados os principais casos.

Várias garagens

Tratar o problema como uma única ou várias garagens não é consequência direta do número de garagens da empresa de transporte público. Se as áreas de atuação de uma empresa com várias garagens não se interceptam, isto é, cada linha está previamente associada a uma única garagem, o problema pode ser dividido em um conjunto de sub-problemas com uma única garagem. Caso contrário, trata-se de um problema de programação de veículos com várias garagens, cada uma com uma dada capacidade. Este é um problema *NP-difícil* (Bertossi et al. 1987) que tem sido

modelado e resolvido de diferentes maneiras, tanto por meio de métodos exatos, quanto por abordagens heurísticas.

Os métodos exatos mais bem sucedidos são baseados em modelos de fluxo em redes, e em suas analogias com a programação inteira. Na literatura, existem dois modelos básicos para o problema. No primeiro, um modelo de arcos orientados leva a um *Problema de Multifluxo*, e no segundo um modelo de caminho orientado leva ao *Problema de Particionamento*. O segundo pode ser obtido aplicando-se a decomposição de Dantzig-Wolfe ao primeiro. Tais problemas são tradicionalmente resolvidos utilizando-se técnicas como a relaxação linear, relaxação lagrangeana, geração de colunas, branch-and-bound, branch-and-price (Bertossi et al. 1987, Carpaneto et al. 1989, Desrosiers et al. 1988, Ribeiro e Soumis 1994, Forbes et al. 1994, Löbel 1998).

O problema pode ser resolvido heurísticamente decompondo-o em subproblemas mais simples, que podem ser resolvidos com algoritmos polinomiais. Duas alternativas de decomposição são:

a) *Programação do veículo corrente* é uma solução simples que produz bons resultados, tendo sido muito utilizada na prática. Como é uma abordagem eficiente e de fácil implementação, é também utilizada para encontrar soluções iniciais de outros problemas. Bodin et al. 1978 descrevem este procedimento assim:
Passo 1. Ordenar as viagens pelo horário de partida e designar a viagem 1 ao veículo 1.
Passo 2. Para $i = 2$ ao total de viagens, se for possível designar a viagem i a um veículo existente, então designe-a ao veículo que envolve o menor custo operacional. Caso contrário, crie um novo veículo e designe a viagem i a ele.

b) *Programação em duas fases* descrita em Bodin et al. (1983), que conta com as seguintes variações:
i) *Primeiro agrupa - depois programa*: o conjunto de viagens é particionado em subconjuntos associados às diferentes garagens. Assim, são resolvidos separadamente vários problemas com uma única garagem.
ii) *Primeiro programa - depois agrupa*: inicialmente um problema com uma garagem fictícia é resolvido, produzindo um bloco de viagens para cada veículo

da frota. Então, cada bloco de viagens é designado a uma dada garagem, de tal maneira que o custo total de viagens ociosas ligando os blocos às respectivas garagens seja minimizado. Adicionalmente, as limitações das garagens são satisfeitas. A segunda estratégia mostra-se mais eficiente do que a primeira, na qual a partição do conjunto de viagens é arbitrária e pode resultar em programações pobres.

Um número grande de trabalhos científicos que abordam este problema e suas possíveis extensões é encontrado na literatura (Dell'Amico et al. 1993, Mesquita e Paixão 1992, Lamatsch 1992, Bertossi et al. 1987, El-Azm 1985, Smith e Wren 1981, Ceder e Stern 1981), sendo portanto, um dos problema de programação de veículos mais estudado na atualidade, seguido pelo problema de programação de veículos com frota mista.

Diferentes tipos de veículos

Ao definir a frota, pode ser necessário considerar diferentes tipos de veículos na operação, como por exemplo mini-ônibus, ônibus padrão e articulados. Isso ocorre devido a fatores tais como a variação da estrutura da demanda, as condições técnicas, a diferenciação no serviço, etc. Do ponto de vista econômico, a capacidade de cada veículo deve corresponder à demanda prevista nas respectivas viagens que ele vai executar. Assim, o *problema de Programação de Veículos de Diferentes Tipos* (ou *com Frota Mista*), é baseado num dado conjunto de viagens , a serem atendidas por diferentes tipos de veículos. O objetivo é determinar uma solução que minimize o custo operacional da frota, associando a cada viagem um veículo com capacidade adequada, e respeitando a limitação do número total de veículos disponível por tipo. Este problema ainda apresenta algumas variantes, dependendo de questões tais como:

- Cada viagem só pode ser executada por um único tipo de veículo ou existe uma ordem de prioridades?
- Os veículos devem sempre retornar à garagem de origem ou é possível fazer rodízio entre as garagens?

Tais problemas podem ser formulados matematicamente, porém com um nível de

complexidade *NP-difícil* (Bertossi et al. 1987), e portanto tem sido atacado por meio de heurísticas.

Costa et al. (1995) apresentam três diferentes fomulações para o problema: como um *Problema de Multifluxo*, como um *Problema de Programação de Veículos com Várias Garagens*, e como um *Problema de Particionamento* com restrições laterais. Para obter um limitante inferior do valor ótimo do problema, foram desenvolvidas três heurísticas: aplicando relaxação linear às duas primeiras formulações e a relaxação lagrangeana para a terceira. Também foi considerada uma heurística para obter um limitante superior para o valor ótimo do problema.

Ceder (1995) desenvolveu uma heurística baseada na *Teoria da Função de Déficit*. A função de déficit utilizada para o problema é uma função escada que aumenta em uma unidade a cada partida e diminui em uma unidade a cada chegada, para um determinado terminal. Esta função pode ser construída para cada terminal, quando se tratar de um sistema multiterminais. Neste caso os diferentes tipos de veículos são listados na ordem decrescente dos seus custo operacioanais. Assim, cada viagem pode ser executada pelo veículo do seu tipo ou de qualquer outro tipo listado acima dele. A heurística começa estabelecendo limitantes inferiores e superiores para o tamanho da frota. Entre estes limitantes, o algoritmo procura por uma solução melhor, aplicando várias propriedades da função de déficit. Entre elas, o algoritmo pode iniciar sua busca pela otimalidade baseada em diferentes critérios: *i*) torca de horários de partida primeiro, inserção de viagens ociosas depois; *ii*) apenas com a inserção de viagens ociosas; *iii*) inserção de viagens ociosas simultaneamente como troca de partidas. Uma das principais vantagens da função de déficit é quanto à visualização natural do processo de otimização. Essa característica permite observar resultados intermediários e avaliá-los enquanto o algoritmo executa as etapas seguintes. A interação entre a tabela de horários, programação dos veículos e o programador é fundamental, permitindo que o programador interfira no processo sempre que julgar conveniente.

Kwan e Rahin (1995) apresentam uma implementação orientada a objetos do algoritmo para programação de veículos *BUSMAN* (Wren e Chamberlain 1988). A base do algoritmo consiste em montar uma programação inicial, usando um determinado número de veículos. Nesta solução são permitidas ligações infactíveis, nas quais

presume-se que um veículo parte do terminal antes mesmo da sua chegada. A partir da solução inicial aplica-se a estratégia de 2-ótimo, que tenta eliminar as ligações infactíveis, testando diferentes possibilidades de troca.

O sistema *HOT* (Daduna e Mojsilovic 1988) usa uma heurística que inclui as restrições do problema na construção do modelo que representa a base do processo combinatório.

Número limitado p de veículos

Em alguns casos, um número fixo de veículos está disponível para operar uma determinada tabela de horários. Nestes casos a programação a ser obtida deve minimizar o número de viagens não executadas. A programação de um número fixo de veículos pode acontecer pelos seguintes motivos:

a) O PBPV é resolvido em duas fases. Na primeira fase é determinado o número mínimo de veículos necessários e na segunda fase é procurado o custo operacional mínimo.

b) Quando uma frota maior que o número mínimo está disponível e todos os veículos devem ser usados.

c) A frota não é grande suficiente para operar as viagens planejadas e é considerada a possibilidade de algumas viagens não serem executadas.

Nos casos a) e b) o problema pode ser resolvido como problema de transporte ou de designação. Para o caso mais geral c), no qual é permitido que algumas viagens não sejam executadas, somente o algoritmo de pseudo-designação pode ser aplicado. Para que o problema seja resolvido pelo método do assinalamento, a rede deve ser expandida, criando-se um nó para cada veículo, assim como arcos que ligam estes às viagens. Assim, o problema fica equivalente a encontrar um emparelhamente com custo mínimo (Bertossi et al. 1987, Daduna e Paixão 1995).

Limitação no tempo de operação

O *Problema de Programação de Veículos com Limitação de Duração* é um problema de programação com a restrição adicional que o tempo de operação de cada veículo não deve ultrapassar um determinado período de tempo (Bodin et al. 1983,

Desrosiers et al. 1988, Freling e Paixão 1995). Neste caso a restrição se deve a questões técnicas, normas trabalhistas, etc.

Periodicidade da programação

Em geral, o problema de programação de veículos é de natureza periódica, no sentido que, após um dado período, como por exemplo 24 horas, todas as viagens devem ser executadas novamente. Na maioria dos casos basta determinar uma programação ótima para um dia, e aplicá-la repetidamente para os demais dias. Normalmente são praticadas três programações diferentes: uma para os dias úteis, uma para os sábados e uma para os domingos (Carraresi e Gallo 1984).

As condições operacionais segundo as quais a programação dos veículos deve ser executada, dimensiona a complexidade do problema. A abordagem matemática adotada depende dos aspectos operacionais a serem incluídos no modelo. Algumas vezes, a simplificação de algumas características da operação reduz consideravelmente a complexidade do modelo matemático. Entretanto, na grande maioria dos casos acima mencionados, as técnicas de resolução acabam recorrendo à resolução do problema de programação de veículos com uma única garagem e um único tipo de veículo, sendo portanto denominado *Problema Básico de Programação de Veículos*, ou simplesmente *Problema de Programação de Veículos - PPV*, tratado a seguir.

Os métodos de resolução do problema de programação de veículos podem ser classificados em três grupos, segundo a utilização ou não de procedimentos heurísticos.

a) Métodos Exatos: utilizam algoritmos que convergem para a solução ótima global. O tempo de processamento destes algoritmos está associado à sua complexidade, tornando-se impraticáveis para problemas *NP-completos*.

b) Métodos Híbridos: combinam algoritmos exatos com procedimentos heurísticos para resolver problemas *NP-completos* em uma razoável tempo de processamento. A utilização destes métodos não garante a obtenção do ótimo global.

c) Métodos Heurísticos: não utilizam qualquer modelo matemático, baseando-se em procedimentos computacionais capazes de gerar soluções satisfatórias sem, no entanto, garantir o ótimo global. Estes métodos são geralmente de fácil implementação e adaptação aos casos particulares.

Abaixo são descritos os principais métodos exatos para o problema, e o método heurístico *BOOST*, que faz parte desta pesquisa.

3.3 A Representação Básica do Problema

Este problema consiste em alocar um conjunto de veículos a uma dada tabela de viagens, de tal maneira que todas as viagens sejam executadas e o custo total seja mínimo. Neste caso a frota é homogênea e existe apenas uma garagem. O modelo matemático para o *Problema Básico de Programação de Veículos* (PPV) pode ser visto como um Problema de Programação Inteira ou como um Problema de Fluxo em Redes. São inúmeros os autores que utilizam os modelos de fluxo em redes para resolver o PPV, como pode ser verificado nos trabalhos de revisão bibliográfica de Bodin et al. (1983), Carraresi e Gallo (1984) e Daduna e Paixão (1995). Isto se deve pela eficiência de tais modelos em representar o problema e na simplicidade de implementação dos algoritmos de fluxo em redes, os quais na sua maioria apresentam complexidade polinomial. Além disso, uma vez consideradas apenas unidades de fluxo inteiras, a solução ótima será necessariamente inteira, como é de se desejar. Tal atributo destes algoritmos evita também problemas de instabilidade numérica, visto que não ocorrem erros de arredondamento (Murty 1992).

Numa representação inicial, cada viagem é simbolizada por um nó e os arcos representam as possibilidades de encadeamento consecutivo das duas viagens nas suas extremidades. A garagem é representada por dois nós: um para a partida e outro para o retorno do veículo à garagem. O nó de partida da garagem está ligado ao início de cada viagem, assim como cada nó final de viagem está ligado ao nó de retorno à garagem. Estes arcos conectam as viagens à garagem, permitindo que qualquer viagem seja o início ou o término de um "bloco de viagens"a ser executado por um determinado veículo. Uma vez que não é conhecido a priori o número mínimo de veículos, atribuímos à garagem um total de veículos disponíveis igual ao total de viagens. Logo, o nó de partida da garagem tem oferta de fluxo igual ao total de viagens e o nó de retorno à garagem tem uma demanda de fluxo de mesmo valor. Isto permite uma solução onde cada veículo executa uma única viagem, ou seja, o pior caso. Na maioria das situações o número mínimo de veículos é inferior

ao número de viagens, havendo um excedente de oferta de fluxo no nó de partida da garagem e uma carência de unidades de fluxo no nó de retorno à garagem. Para atingir o equilíbrio nestes nós, é introduzido um arco ligando-os com custo nulo e capacidade igual ao número total de viagens programadas.

O problema pode então ser definido em termos de fluxo em redes da seguinte maneira. Dado um conjunto de n viagens, $V = \{1, ..., n\}$, cada viagem $i \in V$ é representada por:

- b_i o ponto inicial,
- e_i o ponto final,
- d_i o horário de partida de b_i
- a_i o horário de chegada em e_i.

Para qualquer par de viagens (i, j), t_{ij} representa o *tempo de viagem em trânsito, viagem ociosa* ou *viagem morta* de e_i até b_j. A garagem é representada pelos nós r e s, onde r é o nó de partida da garagem e s é o nó de retorno à garagem, ambos os nós referentes ao mesmo local. Logo, t_{ri} é o tempo de viagem morta da garagem até b_i e t_{is} é o tempo de viagem morta de e_i até à garagem. Um par de viagens (i, j) é dito *compatível* se $t_{ij} \leq d_j - a_i$. Ou seja, quando existe tempo suficiente para que um mesmo veículo execute a viagem i, desloque-se até o ponto inicial da viagem j caso seja necessário, e esteja apto a iniciá-la no horário previsto. O custo de um arco (i, j) ligando as viagens i e j é dado por:

$$c_{ij} = \begin{cases} K_1 t_{ij} + (Custo\ Fixo)/2 & \text{se } i = r \text{ ou } j = s \\ K_1 t_{ij} + K_2(Tempo\ de\ Espera) & \text{se } (i, j) \text{ forem viagens compatíveis,} \\ \infty & \text{caso contrário.} \end{cases} \quad (3.1)$$

o *Custo Fixo* é atribuído à primeira viagem de cada veículo que parte da garagem e à última viagem quando o veículo retorna à garagem. Esta atribuição pode também ser feita unicamente à primeira ou à última viagem do veículo e tem como finalidade minimizar a frota em operação.

O custo variável é computado em função do tempo que o veículo fica ocioso nos terminais, ou enquanto realiza viagens improdutivas entre dois terminais, ou entre

um terminal e a garagem. As constantes K_1 e K_2 são valores definidos pelo usuário que devem trazer informações sobre os diferentes custos associados ao veículo e à tripulação. O *tempo de espera* é dado por $d_j - a_i - t_{ij}$ até um determinado valor. Quando este tempo supera um dado limite, o veículo deve retornar à garagem para uma *parada temporária*. Dell'Amico et al. (1993) adotam os valores 10 e 2 enquanto Kwan e Rahin (1999) adotam 2 e 1 para K_1 e K_2 respectivamente. Como se observa, os valores para os pesos variam de acordo com a situação operacional, porém K_1 é sempre maior do que K_2 em casos reais.

Gavish et al. (1978) decompõem o custo pelo encadeamento de um par de eventos compatíveis da seguinte forma:

$$c_{ij} = \begin{cases} D & \text{se } i = r,\ j = 1, ..., n \\ 0 & \text{se } i \in V \cup \{r\},\ j = s \\ L_{ij} = E_{ij} + K_{ij} + R_{ij} + W_{ij} & i, j = 1, ..., n \end{cases} \tag{3.2}$$

onde $\tilde{D}$ é a economia anual obtida com a redução de um veículo na frota, e D é a parte de $\tilde{D}$ correspondente ao período da programação. E_{ij} é o custo direto relativo à viagem ociosa entre i e j, incluindo custos operacionais e os custos diretos dos motoristas. K_{ij} é uma penalização por desconectar uma ligação existente anteriormente entre as viagens. Esta penalização tem como objetivo reduzir o número de mudanças em relação à programação previamente em operação. R_{ij} é uma penalização por encadear viagens pertencentes a diferentes centros de controle. No caso do sistema de transporte urbano, os centros de controle estão associados às diferentes garagens da empresa operadora. W_{ij} é o custo do tempo de espera da tripulação entre as viagens i e j.

É fácil verificar que se $L_{ij} \geq D$, então não existe economia com o encadeamento das viagens i e j. Neste caso é mais barato iniciar a viagem j com um veículo proveniente da garagem.

Seja $G = (N, A)$ o *grafo direcionado* com nós $N = V \cup \{r, s\}$ e arcos $A = \{(i, j) \mid (i, j)$ é um par de viagens compatíveis $\forall i, j \in V\} \cup \{(r, i), (i, s), \forall i \in V\}$. Para ilustrar esta rede, considere o conjunto de viagens com seus respectivos horários e locais de partida e chegada contidos na tabela (3.1). Por simplicidade, considere os locais de partida e chegada no mesmo terminal A. É fácil verificar que após executar a viagem 1, o mesmo veículo poderá executar a viagem 3 ou a 4, visto que estas viagens têm

Tabela 3.1: Conjunto de 4 viagens com horários e locais de partida e chegada.

Viagem	Partida	Local	Chegada	Local
1	07:00	Term A	08:30	Term A
2	08:00	Term A	08:50	Term A
3	09:00	Term A	11:00	Term A
4	09:30	Term A	10:00	Term A

seus horários de partida posteriores ao horário de chegada da viagem 1. Como todas as viagens estão baseadas no mesmo terminal, não existem viagens mortas. Logo são incluídos na rede os arcos $(1, 3)$ e $(1, 4)$ ligando a viagem 1 às viagens 3 e 4, com tempos de espera no terminal de 30 e 60 minutos, respectivamente. Os arcos que partem e chegam à garagem apresentam além do custo fixo, o custo operacional dado pelo tempo de viagem morta da garagem até o terminal e do terminal até a garagem, tempos estes que podem diferir, dependendo do sentido da viagem. A figura (3.1) é a representação gráfica da rede G para as viagens na tabela (3.1). A

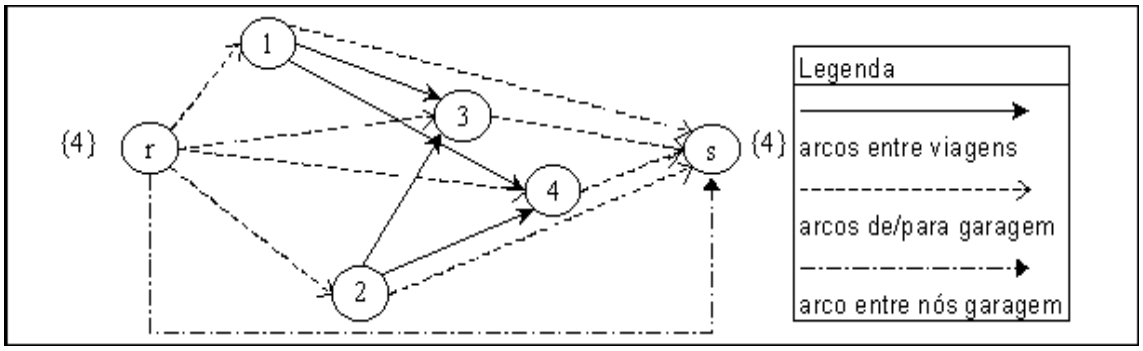

Figura 3.1: Rede genérica para o PPV da tabela (3.1).

rede apresentada na figura (3.1) é uma primeira representação para o problema. Na verdade os algoritmos de resolução não são aplicados a esta rede, mas a uma outra rede que deriva desta. O problema está no fato de que cada nó viagem exige que exatamente uma unidade de fluxo passe por ele. Para que esta característica seja satisfeita, cada nó viagem deve ser desmembrado em dois nós, ligados por um arco cujo fluxo seja obrigatóriamente igual a um. Esta transformação na rede é descrita a seguir.

3.4 O PPV como Problema de Fluxo com Custo Mínimo

Abordar o PPV como um Problema de Fluxo com Custo Mínimo é a forma mais genérica, sendo que qualquer outra formulação pode ser vista como um problema deste tipo. Cada viagem é representada por dois nós: um para o início e outro para o final da viagem. O nó início de viagem consome uma unidade de fluxo enquanto o nó fim de viagem produz uma unidade de fluxo. Caso seja necessária a formação de ciclos, o arco relacionado à garagem deve ser conectado no sentido (s, r).

Na formulação do PPV como um problema de fluxo com custo mínimo, proposta por Dantzig e Fulkerson (1954), cada nó referente à viagem i na figura (3.1) é dividido em dois nós, i' para o início e i'' para o final da viagem. Estes nós são ligados pelo arco (i', i'') com custo nulo e pelo qual deve necessariamente passar uma unidade de fluxo, garantindo assim a execução da viagem. Para forçar a passagem de fluxo no arco (i', i'') basta assumir que o limitante inferior l e o limitante superior u deste arco sejam iguais a 1. Esta duplicação é mostrada na figura (3.2). Para todos os pares

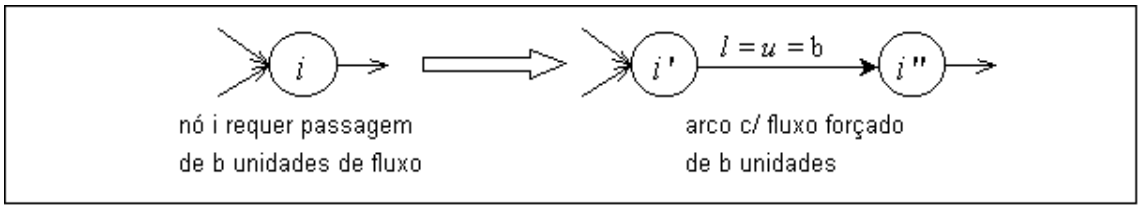

Figura 3.2: Transformação que força um fluxo de b unidades pelo nó i.

(i, j) de viagens compatíveis, é incluído na rede o arco (i'', j') com custo c_{ij} dado pela expressão (3.1), ou (3.2). O nó de partida da garagem é ligado a cada nó início de viagem, e cada nó final de viagem é conectado ao nó de retorno à garagem. Existem duas possibilidades para o arco que liga os nós partida e chegada à garagem. Se este arco estiver no sentido (r, s), então estes nós devem apresentar respectivamente oferta e demanda iguais ao número de viagens. Se o arco estiver no outro (s, r) (dito arco de retorno) tem-se uma rede cíclica. Neste caso todos os nós da rede apresentam demanda nula. Esta segunda possibilidade de ligação dos nós da garagem transforma o problema de fluxo com custo mínimo em um problema de circulação mínima, permitindo aplicar diretamente algoritmos que operam em redes cíclicas, como é o

caso do algoritmo *Out-of-Kilter* (Fulkerson 1961). Em ambos os casos o arco que liga os nós da garagem deve ter custo nulo e capacidade maior ou igual ao número de viagens. A figura (3.3) é um exemplo deste tipo de rede para as viagens da tabela (3.1). Considerando a rede dada por $G = \{N, A\}$, onde $N = \{r, s\} \cup \{i', i'' \ \ \forall i \in V\}$

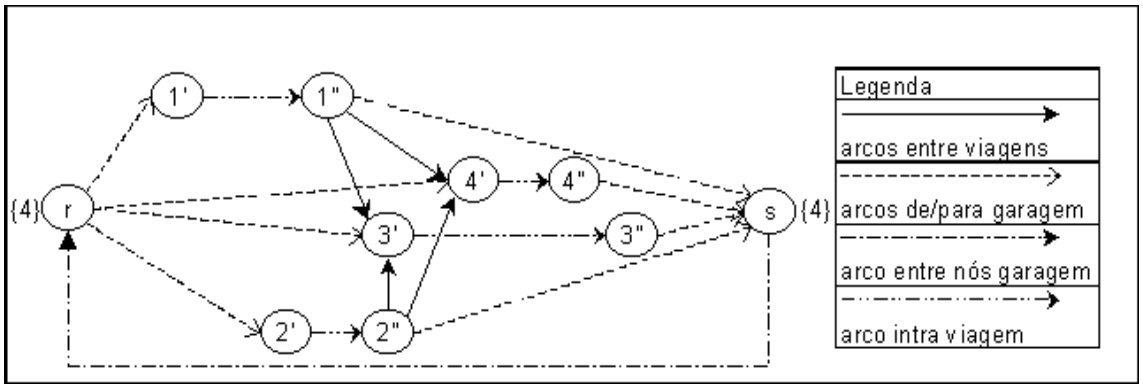

Figura 3.3: Representação do PPV como fluxo com custo mínimo.

e $A = \{(i', i''), (r, i'), (i'', s), \forall i \in V\} \cup \{(i'', j'),\ \forall (i, j)$ par de viagens compatíveis$\} \cup \{(s, r)\}$, o problema de fluxo com custo mínimo para o PPV é dado por:

$$\text{Min} \sum_{(i,j)\in A} c_{ij} x_{ij} \qquad \text{sujeito a} \tag{3.3}$$

$$\sum_{j\in N} x_{ij} - \sum_{j\in N} x_{ji} = 0 \qquad \forall j \in N \tag{3.4}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i, j) \in A - (s, r) \tag{3.5}$$

onde $x_{ij} = 1$ se a viagem j for executada imediatamente após a viagem i e $x_{ij} = 0$ caso contrário. A expressão (3.3) minimiza o custo total, a expressão (3.4) diz respeito ao princípio de conservação de fluxo, enquanto a restrição (3.5) garante os valores zero ou um para o fluxo nos arcos entre viagens compatíveis. Este problema pode ser resolvido com o algoritmo Simplex especializado para redes (Dantzig 1963).

Bokinge e Hasselström (1980) abordam o PPV admitindo flexibilidade nos tempos de chegada e partida das viagens. Os autores formulam o problema como um problema de fluxo com custo mínimo e resolvem-no em duas etapas, utilizando sempre o algoritmo simplex para redes. Na primeira etapa é considerado apenas o *kernel* da cada viagem, determinando assim qual é o horário de maior demanda de veículos e, consequentemente, determinando o limitante inferior para o número de veículos. O *kernel* de uma viagem i é a viagem k_i que tem seu início igual ao início da

viagem ocorrida com maior atraso no seu histórico. O término da viagem k_i coincide com a chegada mais cedo ocorrida com a viagem i. A solução deste problema fornece um limitante inferior para o número de veículos e uma solução inicial para o problema. A segunda fase do método tenta melhorar os blocos de viagens de cada veículo, mudando os tempos de partida das viagens flexíveis, de tal maneira que o tempo de viagens mortas e o tempo de espera entre viagens sucessivas sejam reduzidos. A solução obtida após este processo é comparada com a solução anterior e o método é interrompido quando duas soluções consecutivas apresentarem as mesmas características. O método apresentado tem como objetivos minimizar o número de veículos, minimizar o tempo que os veículos ficam fora da garagem e realizar o menor número possível de alterações na programação corrente.

Para reduzir o tamanho da rede que representa o problema, são criadas "garagens imaginárias" a cada sub-intervalo de tempo (30 minutos) do período a ser programado. Estas garagens intermediárias permitem eliminar os arcos que ligam viagens com longo tempo de espera entre elas, e por isso é dita *técnica de eliminação de arcos*. Ocorre, no entanto, que qualquer veículo que retornar temporariamente à garagem no meio do sub-intervalo deverá permanecer mais tempo do que o "tempo mínimo de garagem" requerido na prática.

A rede apresentada na figura (3.3) trata o arco garagem no sentido (s, r); no entanto, ao adotar o sentido oposto (r, s), a rede resultante pode ser vista como a representação de um problema de transporte equivalente ao de designação, tratados a seguir.

3.5 O PPV como Problema de Designação

Conforme descrito na seção (2.5), o problema de designação está associados a dois conjuntos disjuntos de nós N_1 e N_2 com mesma cardinalidade e um conjunto de arcos $A \subseteq N_1 \times N_2$ representando as possibilidades de associação dos nós. Nestes problema deseja-se associar cada elemento de N_1 a um único elemento de N_2, tal que o custo total seja mínimo.

Em problemas de programação de veículos os conjuntos de nós N_1 e N_2 são definidos

de acordo com a oferta e demanda de fluxo, agrupando de um lado os nós com oferta de fluxo, e do outro os nós com demanda de fluxo. Assim, a rede bipartida tem N_1 dado pelos nós de chegada i'' e N_2 pelos nós de partida das viagens i'. Os arcos da rede ligam cada nó do primeiro conjunto a cada um dos nós do segundo conjunto, ou seja $A = V \times V$, e são divididos em dois subconjuntos:

i) A_1, dos arcos que ligam duas viagem compatível, com custo dado por (3.1), e

ii) A_2, dos arcos que ligam viagens incompatíveis, cujo custo é dado por $c_{i,r} + c_{s,j}$ acrescido do custo fixo incorrido pela utilização de um veículo.

Os arcos do subconjunto A_1 que fizerem parte da solução indicam o encadeamento de duas viagem, enquanto os arcos do subconjunto A_2 na solução indicam o sequenciamento das viagens ociosas relacionadas à garagem e à utilização de um veículo. A garagem não é representada explicitamente, figurando apenas os nós referentes às viagens.

Considerando a rede bipartida dada por $G = (N_1, N_2, A)$ na figura (3.4), o modelo de pseudo-designação para o PPV é dado pelas equações (3.6) a (3.9). Para o exemplo da tabela (3.1), a rede associada a este modelo é mostrada na figura (3.4).

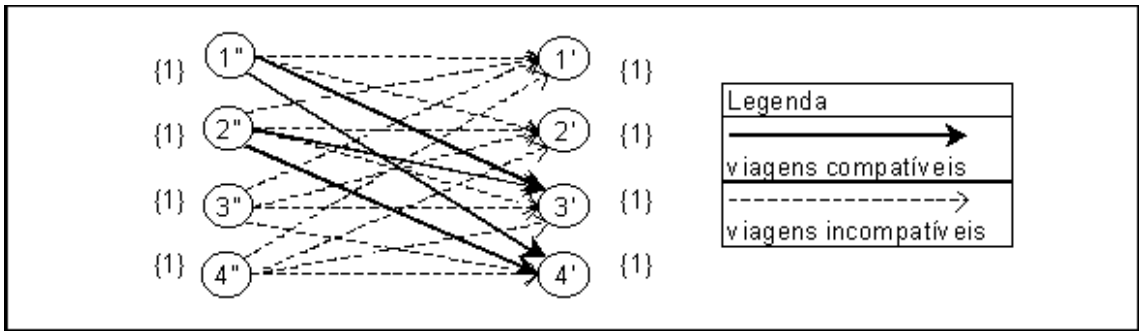

Figura 3.4: Representação do PPV como Problema de Designação.

$$\text{Min} \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \qquad \text{sujeito a} \tag{3.6}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \text{ para } i = 1, ..., n \tag{3.7}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \text{ para } j = 1, ..., n \tag{3.8}$$

$$x_{ij} \in \{0, 1\} \text{ para todo } i, j \in V. \tag{3.9}$$

Segundo Bertossi et al. (1987) uma definição apropriada do custo c_{ij} nos arcos transforma o PPV num *Problema de Emparelhamento Perfeito com Custo Mínimo*, ou seja, em um *Problema de Designação* (Ahuja et al. 1993). Se $c_{ij} = 0,\ \forall (i,j) \in A_1$ e $c_{ij} = 1,\ \forall (i,j) \in A_2$, então é possível encontrar a frota mínima para o problema. Se o custo c_{ij} representar os custos reais de viagem morta e de tempo parado no terminal como em (3.2), então o problema estará minimizando o custo operacional. Também é possível utilizar uma combinação destes custos para minimizar o custo total.

O problema da programação de veículos com uma única garagem é formulado como um *Problema de Emparelhamento Capacitado*. Este modelo é parecido com o modelo de pseudo-designação, sendo que os nós garagem são nós de transbordo, ou seja, não consomem nem produzem unidades de fluxo. Para que o número de veículos utilizado na programação seja limitado, o arco que liga os nós garagem é colocado no sentido (s, r) e tem capacidade igual ao número máximo de veículos permitido. Os autores mostram que para uma estrutura de custo específica, este problema pode ser resolvido como um problema de fluxo com custo mínimo, garantindo assim uma resolução em tempo polinomial.

Para formular o problema de programação de veículos com várias garagens, os autores utilizam a idéia do emparelhamento com multifluxo e demonstram que este é um problema *NP-difícil*. Para resolver tal problema é apresentada uma heurística que aplica a relaxação lagrangeana a uma série de problemas de emparelhamento simples.

Hoffstadt (1981) utiliza uma heurística de encadeamento de viagens para determinar o número mínimo de veículos. Posteriormente, para minimizar o custo operacional é aplicado o algoritmo húngaro (Kuhn 1956) ao respectivo problema de Designação.

Esta formulação produz uma rede com número de arcos maior do que o necessário, visto que todas as viagens estão associadas entre sí. Na verdade, uma viagem só estará associada às viagens cujos horários de partida são maiores do que seu horário de chegada, acrescido de um tempo de preparação.

3.6 O PPV como Problema de Pseudo-Designação

Uma maneira de reduzir o tamanho da rede no modelo de designação consiste em acrescentar um nó de partida e um nó de retorno à garagem, assim como é feito no modelo de fluxo com custo mínimo. Porém, com esta modificação a solução deixa de apresentar as características de um problema de designação. A solução apresentará mais do que uma unidade de fluxo partida da garagem, alimentando diferentes inícios de viagem, o que infringe o conceito de designação. Neste caso tem-se um problema de designação desequilibrado, também denominado *Problema de Pseudo-Designação.*

O problema de pseudo-designação pode ser visto como um problema clássico de transporte, no qual os nós origem de fluxo são o nó de partida da garagem e os nós de término de viagem. E os destinos são os nós de início de viagem e o nó de retorno à garagem. Os algoritmos para o problema de transporte são utilizados e adaptados para resolver o problema de designação e pseudo-designação (Gavish et al. 1978, Gavish e Shlifer 1978, Paixão e Branco 1987).

O modelo de pseudo-designação considera dois conjuntos disjuntos de nós, ligando cada nó de um conjunto a pelo menos um nó de outro conjunto, tal que o custo total das ligações seja mínimo. Portanto, assim como no modelo de fluxo com custo mínimo, para cada viagem $i \in V$ são definidos os nós i' e i''. A garagem é representada pelo nó de partida r e pelo nó de retorno s, sendo que o nó de partida disponibiliza n veículos, enquanto o nó de retorno requer n veículos, visto que após o horizonte de programação, todos os veículos devem retornar à garagem.

Os nós são então divididos em dois conjuntos disjuntos definidos por $N_1 = \{r\} \cup \{i''\ \forall i \in V\}$ dos nós com oferta de fluxo e $N_2 = \{s\} \cup \{i'\ ,\ \forall i \in V\}$ dos nós com demanda de fluxo. O conjunto de arcos ligando os nós de N_1 a N_2 é definido por $A = \{(r, i'),\ \forall i \in V\} \cup \{(i'', j'),\ \forall (i, j)$ par de viagens compatíveis$\} \cup \{(i'', s),\ \forall i \in V\}$. O primeiro subconjunto de A liga a garagem até o início das viagens, o segundo liga dos pares de viagens compatíveis e no terceiro subconjunto, o final de cada viagem é ligado ao retorno à garagem. Assim tem-se a rede bipartida $G^{pd} = (N_1, N_2, A)$.

Um exemplo deste tipo de rede é mostrado na figura (3.5), onde os valores entre

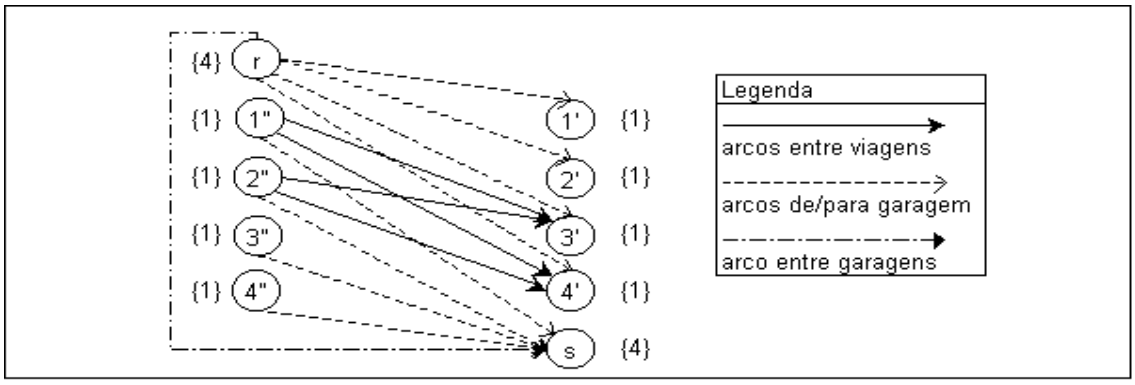

Figura 3.5: Rede G^{pd} para o PPV como um Problema de Pseudo-Designação.

chaves representam as respectivas ofertas e demandas de fluxo de acordo com o tipo de nó da rede, e o custo c_{ij} no arco $(i,j) \in A$ é dado pela expressão (3.1). O arco que liga as garagens tem custo nulo e capacidade n. A formulação matemática para este Problema de Pseudo-Designação é dada por:

$$\text{Min} \sum_{(i,j)\in A} c_{ij}x_{ij} \qquad \text{sujeito a} \tag{3.10}$$

$$\sum_{j:(i,j)\in A} x_{ij} = 1, \qquad \forall\ i \in N_1 - \{r\} \tag{3.11}$$

$$\sum_{i:(i,j)\in A} x_{ij} = 1, \qquad \forall\ j \in N_2 - \{s\} \tag{3.12}$$

$$\sum_{j:(i,j)\in A} x_{sj} = n \tag{3.13}$$

$$\sum_{i:(i,j)\in A} x_{it} = n \tag{3.14}$$

$$x_{ij} \in \mathbf{N} \qquad \text{para todo } (i,j) \in A \tag{3.15}$$

onde a variável de decisão $x_{ij} = 1$ se a viagem j for executada pelo mesmo veículo imediatamente após a execução da viagem i, e $x_{ij} = 0$ caso contrário. A expressão (3.10) minimiza o custo total, as restrições (3.11) e (3.12) asseguram que cada viagem sejá executada uma única vez, enquanto as restrições (3.13) e (3.14) estão relacionadas com as partidas e chegadas dos veículos ao depósito. A restrição final (3.15) garante a integralidade das variáveis.

A partir da solução deste modelo é possível montar os blocos de viagens para os veículos. Nesta rede, cada caminho que sai de r e chega a s corresponde a um bloco de viagens a ser executado por um dado veículo. O conjunto de todos os caminhos de r para s fornece a programação completa com o menor número de veículos e

custo operacional.

Gavish et al. (1978) formulam o PPV como um problema de transportes, que tem como objetivos primários minimizar o número de veículos utilizados durante o horário de pico e minimizar o tempo de viagem morta no horário de entre-pico. O objetivo secundário é minimizar as mudanças em relação à programação já existente.

O problema é dividido em dois estágios, sendo que na primeira etapa as viagens são sequenciadas, formando a programação dos veículos. Cada seqüência é iniciada em uma garagem, contendo viagens de uma ou mais rotas diferentes, com tempos de espera e/ou viagens mortas, e terminando na mesma garagem. Estas seqüências são construídas tentando maximizar as ligações (minimizar o número de veículos) e minimizar os tempos de espera nos terminais e de viagens mortas (minimizar o custo operacional). Estas seqüências são construídas pela resolução de um problema clássico de transportes, onde o custo de cada ligação entre duas viagens é dado pela expressão (3.2). Para resolver o problema de transporte, é utilizado o algoritmo primal simplex para transportes.

Paixão e Branco (1987) apresentam o modelo de transportes e o modelo de designação para o problema de programação de veículos com uma única garagem. Embora sejam equivalentes, tais modelos sugerem métodos diferentes de solução. Os autores destacam que o algoritmo especializado para resolver o problema de transporte apresentado em Gavish et al. (1978) realiza um grande número de pivoteamentos degenerados, ou seja, apresentam o *fenômeno do pivoteamento nulo*, fenômeno este que consome tempo de processamento sem que haja melhoria na função objetivo. Por outro lado, qualquer versão do método húngaro (Kuhn 1956), que faz uma atualização da matriz de custo, exige o armazenamento completo desta matriz, reduzindo a eficiência do método. Para superar tais dificuldades, os autores fazem uma adaptação do método húngaro para o problema de programação de veículos, armazenando apenas os custos associados às ligações factíveis.

3.7 A Redução da Rede

Freling et al. (1995) formulam o PPV como um problema de pseudo-designação, e mostram que o mesmo pode ser formulado como um problema de designação com restrições de desigualdade. No entanto, o maior problema na resolução de casos reais é quanto ao elevado número de arcos, o que leva a uma degenaração no tempo de processamento da solução ótima. Para reduzir a dimensão da rede, os autores apresentam três técnicas distintas.

a) Uma adaptação da técnica apresentada por Bokinge e Hasselström (1980) na qual uma nova garagem fictícia é criada a cada sub-intervalo de tempo. Freling et al. (1995) criam uma nova garagem fictícia sempre que um novo arco de saída da garagem for encontrado.

b) Um método de resolução que tira proveito da estrutura especial do custo nos arcos da rede. Este método consiste em construir os blocos de viagens numa primeira fase e posteriormente combinar tais blocos, formando assim a programação dos veículos.

c) Uma abordagem que gera parte dos arcos da rede, na medida em que for necessário. Esta técnica é mais geral do que as anteriores, pois permite incluir o tempo mínimo de estacionamento temporário na garagem no cálculo do custo dos arcos. A geração de arcos é uma técnica similar à técnica de geração de colunas na programação linear, daí o termo *geração de arcos* para tal abordagem.

Os autores propõem uma adaptação do algoritmo do leilão para resolver o problema de pseudo-designação. O algoritmo do leilão é um algoritmo primal-dual para problemas de fluxo em redes desenvolvido basicamente para processamento paralelo, tendo-se mostrado muito eficiente também no processamento seqüencial (Bertsekas 1991). Os autores aplicam a técnica de ε-scaling às condições de folgas complementares, que consiste em aplicar o algoritmo do leilão várias vezes, iniciando com valor grande para e reduzindo-o até um determinado limite pré-determinado.

3.8 Principais Trabalhos de Revisão

Três trabalhos de revisão dos modelos de fluxo em redes para o problema de programação de veículos devem ser destacados:

Bodin et al. (1983) apresentam uma classificação completa a respeito dos vários problemas de roteamento, programação , e roteamento e programação integrados. O problema de roteamento consite em determinar um conjunto de ciclos centralizados em um depósito, tal que cada nó da rede seja visitado uma única vez, minimizando o custo total de transporte. Se não houver qualquer restrição quanto ao horário da visita, nem quanto à ordem de precedência entre as visitas, então este é um problema de roteamento puro. No entanto, se houver um horário pré-determinado para cada visita, o problema será o de programação de veículos. Fica assim caracterizado o aspecto temporal na diferenciação entre problemas de roteamento e de programação de veículos.

Os autores também abordam a questão da complexidade algoritmica dos métodos de resolução. Um algoritmo com complexidade polonomial para um dado problema é um procedimento computacional cujo limitante do número de operações aumenta polinomialmente com o tamanho do problema. A classe de todos os problemas para os quais se conhece um algoritmo polinomial é denotada por P. Problemas nesta classe P geralmente podem ser resolvidos eficientemente. Em contrapartida, problemas combinatoriais para os quais ainda não se conhece um algoritmo polinomial são ditos *NP-difícil*. Portanto, o esforço computacional necessário para resolver problemas *NP-difícil* cresce exponecialmente com o tamanho do problema, tornando sua resolução impraticável. Daí a importância de se encontrar componentes ou subproblemas do problema *NP-difícil* pertencentes à classe P, que podem ser úteis na resolução do problema original. Cuidados devem ser tomados, pois uma pequena mudança em um problema da classe P pode remetê-lo à classe *NP-difícil*.

Outro importante conceito colocado neste trabalho é o de uma *heurística*. Uma heurística é definida como "um procedimento que se basea na estrutura do problema e na experiência prática para encontrar uma solução factível ou quase ótima". Uma heurística é considerada eficiente, se ela encontra uma solução que se aproxima do ótimo. Em muitos casos a heurística também encontra o ótimo.

São abordados no trabalho os seguintes problemas associados ao transporte público:

- Problema de Programação de Veículos com uma Única Garagem,
- Problema de Programação de Veículos com Várias Garagens,
- Problema de Programação de Veículos de Diferentes Tipos,
- Problema de Programação de Veículos com Limitação no Tempo de Operação
- Problema de Programação de Veículos e Tripulação, e
- Problema do Rodízio de Turnos da Tripulaçao.

Para cada um dos problemas listados acima, os autores apresentam uma modelagem matemática, apontam os problemas que podem surgir, assim como discutem os possíveis métodos de resolução.

Carraresi e Gallo (1984) organizam em uma linguagem unificada os mais relevantes modelos de fluxo em redes que representam alguns problemas do transporte público. Neste contexto são formulados os seguintes problemas: *i*) de programação de veículos, *ii*) de programação da tripulação e *iii*) do rodízio da tripulação. O problema de programação de veículos com uma única garagem é formulado como um problema de designação, ao qual posteriormente são adicionados dois nós relacionados à garagem, para que haja uma diminuição no número de arcos da rede. O problema com várias garagens é resolvido com sua decomposição em problemas mais simples, ou seja, utilizando as heurísticas agrupa/programa ou programa/agrupa (Bodin 1983).

Daduna e Paixão (1995) descrevem inicialmente várias restrições adicionais que devem ser levadas em consideração quando são estudados casos reais. O problema de programação de veículos com uma única garagem é formulado como: *i*) problema de fluxo com custo mínimo, que no caso se resume a um problema de circulação, *ii*) problema de transporte, que também pode ser visto como um problema de pseudo-designação e *iii*) problema de designação.

O problema de programação de veículos com um número fixo de veículos pode ocorrer nos seguintes casos.

a) O problema de programação de veículos é resolvido em duas fases. Na primeira fase é determinado o número mínimo p de veículos. Na segunda fase é minimizado o custo operacional na utilização destes p veículos.

b) Quando se deseja utilizar um determinado número p de veículos, o qual deve ser maior ou igual ao número mínimo necessário.

c) O número p de veículos disponíveis é inferior ao número mínimo necessário.

Nos casos a) e b) o problema de programação de p veículos poderá ser resolvido pelos modelos mencionados acima, com exceção do método de designação. Para o caso c), quando é permitido que alguma viagem não seja executada, são apresentados outros dois modelos. O modelo de emparelhamento pode ser utilizado criando-se p nós garagem. Neste caso, cada um dos p nós de partida da garagem é ligado ao início das viagens e os nós final de viagem são ligados a cada um dos p nós de retorno à garagem. Visto que algumas viagens não serão executadas, para cada viagem cria-se um arco ligando seu nó final ao seu nó início de viagem. Este arco tem como custo o valor da penalização pela não execução da viagem.

Uma maneira mais simples é considerar o modelo de pseudo-designação atribuindo aos nós garagem oferta e demanda igual ao número p de veículos disponíveis. Analogamente ao modelo de emparelhamento são criados os arcos ligando o final de cada viagem ao seu início com custo de penalização pela não execução da viagem.

A tabela abaixo contém os principais autores que estudaram os modelos descritos acima, além de mencionar a estratégia utilizada para resolver o modelo proposto.

Tabela 3.2: Principais métodos de resolução do problema de programação de veículos.

AUTORES	MODELO	SOLUÇÃO
Dantzig e Furkerson 1954	Fluxo com custo mínimo	Simplex para redes
Gavish et al. 1978	Transporte	Simplex para redes
Hoffstadt 1981	Designação	Algoritmo Húngaro
Paixao e Branco 1987	Pseudo-designação	Alg. Húngaro adaptado
Bertossi et al. 1987	Emparelhamento	Relaxação Lagrangeana

3.9 Outros Métodos Exatos

Martin-Löf (1970) resolve o problema de programação de veículos considerando viagens com flexibilidade nos horários de partida. Este problema é formulado como um problema de programação linear inteira que não está associado a um problema de fluxo em redes e para o qual não existem algoritmos eficientes. Para resolvê-lo o autor utiliza um método simples de busca do tipo *branch-and-bound*. Entretanto, este método não é aplicável a problemas de grande porte.

Stern e Cender (1981) utilizam a de função de déficit associada aos terminais para definir o número mínimo de veículos necessários para executar um conjunto de fixo de viagens. A função de déficit é uma função do tipo escada que varia no tempo, tendo seu valor aumentado de uma unidade no horário de partida de uma viagem e diminuído de uma unidade no horário de chegada de uma viagem. Em um sistema multiterminal, uma função deste tipo é construída para cada terminal. Para construir tais funções são necessários apenas os horários e locais de partida e chegada das viagens. Agrupando-se as funções de déficit dos diversos terminais do sistema, é possível obter uma função geral de déficit que fornece o número de viagens em operação simultânea. Os autores mostram que o número veículos que deve iniciar suas programações em um determinado terminal coincide com o valor máximo da respectiva função de déficit. Este valor máximo representa também o número mínimo de veículos que deve estar disponível no terminal para servir a todas as viagens que partem deste ponto, sem que haja uma realocação de veículos entre os terminais. Desta maneira é formulado o teorema que determina o número mínimo de veículos.

Os autores descrevem um algoritmo que inicialmente faz o sequenciamento das viagens sem a realocação dos veículos. Posteriormente, são incluídas as viagens mortas, de tal maneira que o número de veículos necessários seja reduzido. Este algoritmo permite uma participação ativa do usuário, possibilitando tirar proveito de conhecimentos prévios da situação prática.

3.10 O Método Heurístico *BOOST*

O sistema *BOOST* foi utilizado neste trabalho para que as soluções fornecidas pelos métodos exatos estudados pudessem ser comparadas com as soluções de um sistema heurístico voltado para as necessidades práticas da operação do transporte público. Sendo assim, foi feito um levantamento histórico do sistema *BOOST*, que vai desde a sua versão inicial, o *VAMPIRES*, até à versão final, o *BOOST*, utilizado neste trabalho.

Atualmente o *BOOST* é considerado o mais importantes sistema de programação de veículos do Reino Unido, e tem servido como referência entre os pesquisadores da área. Isto se deve pela tradição do grupo responsável pelo seu desenvolvimento, pelo poder de resolução de problemas mais complexos que surgem na prática, e pela larga utilização deste sistema por companhias de transporte urbano do Reino Unido e fora dele.

O *VAMPIRES* é a primeira versão de uma série de sistemas heurísticos. Ele foi desenvolvido tendo como base o programa de programação de locomotivas descrito por Wolfenden e Wren (1966), e foi utilizado pela primeira vez para programar veículos em 1970. A evolução deste sistema pode ser acompanhada através dos trabalhos publicados a seu respeito.

Wren (1972) apresenta a primeira versão do método heurístico *VAMPIRES* para programação de veículos, que serviu como base para o desenvolvimento dos sistemas seguintes, até chegar à última versão que é o *BOOST*. Nesta versão é considerado um único tipo de veículo e uma única garagem. As viagens têm horários fixos de partida, embora a solução informe quais são as viagens que devam ter seus horários de partida alterados para que haja uma economia no número de veículos. A estratégia é partir de uma solução inicial simplificada, construída tendo como base um determinado número de veículos. A partir daí é aplicada uma heurística do tipo 2-ótimo que tenta melhorar a solução. Caso haja algum "conflito"na programação resultante, o processo é repetido e uma nova solução é encontrada.

Esquema Geral do Método: as viagens são ordenadas de acordo com seus horários de partida e as n primeiras viagens são atribuídas aos n veículos na garagem.

O número n de veículos pode ser fornecido pelo usuário ou calculado automaticamente pelo sistema. Verificando cada uma destas n viagens, o sistema procura a próxima viagem que pode ser executada pelo mesmo veículo e atribui esta viagem ao veículo. Se nenhuma viagem posterior puder ser executada pelo veículo, ele retorna à garagem. É possível que a programação inicial tenha algumas ligações infactíveis (conflitantes), isto é, o ônibus chegue num determinado terminal depois do seu suposto horário de partida. Neste caso, a programação é refinada através de um processo iterativo, em que cada par de ligação é quebrado e religado no sentido oposto caso haja uma redução na infactibilidade total e/ou no custo da programação. Um exemplo de religação de um par de viagens no sentido oposto é apresentado na figura (3.6). O sistema retorna a melhor solução encontrada considerando o número de veículos escolhido anteriormente. Caso este número seja muito baixo, são apontadas as viagens cujos horários devam ser alterados, tal que a programação com o número de veículos especificado seja factível. Na próxima iteração, o número de veículos é aumentado de uma unidade ou de acordo com a opção do usuário e o sistema volta a aplicar o método de 2-ótimo produzindo uma nova solução. Este processo continua até que todas as ligações infactíveis sejam eliminadas.

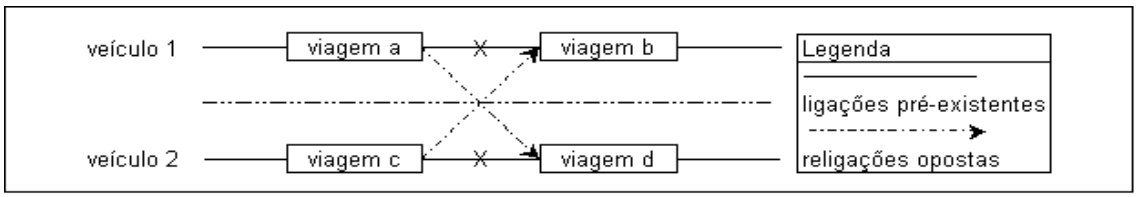

Figura 3.6: Religação oposta de um par de ligações pré-existentes.

Na versão apresentado por Manington e Wren (1975) o sistema já trata da programação de veículos com mais de uma garagem. Isto é feito considerando a primeira e a última viagens do veículo ligadas à garagem, da mesma forma como as outras ligações durante o processo de otimização. Ocorre, no entanto, que sempre que um par de ligações for trocado, todo o trabalho dos dois veículos é alterado de tal maneira que a melhor garagem para os veículos pode ser mudada também. Logo, os efeitos de qualquer troca de ligações são examinados antes de serem aceitos. Smith e Wren (1981) introduzem uma lista de parâmetros aos dados de entrada, os quais permitem uma flexibilização nos horários de partida das viagens, além de iniciar uma estratégia para resolver o problema de programação com vários tipos de veículo. O

objetivo nesta versão do sistema é dividido em dois. O primeiro objetivo se preocupa em minimizar o número de veículos e o segundo objetivo é uma combinação dos seguintes objetivos secundários:

1. os veículos que trabalham somente durante o horário de pico devem ter um tempo de trabalho tão próximo quanto possível de um valor pré-determinado,
2. as viagens não freqüentes, como viagens escolares ou relacionadas a hospitais, devem ser atribuídas ao início ou ao final da jornada de trabalho de um veículo,
3. a programação resultante deve ser tão parecida quanto possível à programação anteriormente em operação, e
4. caso haja paradas temporárias na garagem, elas devem ser tão longas quanto possível.

Os três principais melhoramentos desta versão do sistema em relação à versão anterior são:

1. gerar automaticamente as paradas temporárias dos veículos na garagem, caso o veículo tenha um longo período de espera nos terminais,
2. gerar automaticamente novos horários de partida para as viagens com ligações infactíveis e,
3. programar diferentes tipos de veículos ao mesmo tempo.

Este trabalho apresenta vários casos nos quais o sistema gera soluções mais econômicas do que as programações vigentes anteriormente.

O trabalho de Wren e Chamberlain (1988) incorpora a programação da tripulação ao sistema, que é então denominado de *BUSMAN*. A componente que faz a programação dos veículos, denominada *BUSPLAN*, gera os elementos necessários para que a programação da tripulação seja produzida. Esta versão é executável em microcomputadores, tornando o sistema mais acessível às companhias de transporte público. Os melhoramentos apresentados no trabalho de Chamberlain e Wren (1992) ocorrem basicamente na estratégia de resolução do problema de programação da tripulação, que então se torna a principal preocupação do grupo de pesquisa.

Kwan e Rahin (1995) apresentam os módulos incorporados à componente BUSPLAN do sistema, que permitem uma certa interatividade com o usuário. São apresentadas ferramentas semi-automáticas para coordenar a geração de novos horários das viagens com ligação infactível, além de melhoramentos nas heurísticas que tratam da programação com diferentes tipos de veículos. Este trabalho descreve a modelagem do sistema a ser implementado em linguagem orientada a objetos.

Kwan e Rahin (1999) apresentam as características da implementação do *BOOST*, que é a versão orientada a objetos do sistema anterior, o *BUSPLAN*. Este novo sistema tem uma interface gráfica que facilita a interação com o usuário.

3.11 Conclusões

Durante esta revisão bibliográfica foram levantadas as principais formulações e métodos de resolução para o Problema de Programação de Veículos, tendo se destacado as seguintes características nos trabalhos que abordam o tema.

1. Os métodos de redução da rede têm sido pouco explorados, embora alguns trabalhos estejam se preocupado com este aspecto.

2. Existe uma grande preocupação com o algoritmo de solução, deixando uma lacuna no que se refere às dificuldades encontradas na aplicação dos modelos matemáticos a casos reais.

3. Exite uma carência de pesquisas focadas na formulação do problema, indicando formas alternativas para incluir no modelo restrições adicionais que surgem na prática.

Consequentemente, esta pesquisa foi direcionada no sentido de:

1. aprofundar os conhecimentos em relação às técnincas de redução da rede, e

2. realizar estudos de caso para detectar as características inerentes à realidade brasileira, propor soluções para os problemas estudados, e verificar a aplicabilidade destas soluções.

A seguir são apresentados em detalhes os métodos de redução da rede abordados neste trabalho.

Capítulo 4

Métodos de Redução da Rede

4.1 Introdução

Embora o problema de programação de veículos com uma única garagem pertença à classe dos problemas P (Bertossi et al. 1987), o número de arcos cresce sobremaneira com o aumento no número de nós. Assim, a rede que representa o PPV contém um grande número de arcos para casos reais, o que compromete o tempo de processamento dos algoritmos de otimização. Daí a necessidade de descartar arcos sem a perda da otimalidade. Isto pode ser feito eliminando os arcos que certamente não tomarão parte da solução ótima, ou substituindo parte deles por outros já existentes na rede. Neste sentido é feita a seguinte classificação dos arcos da rede:

a) *arcos longos:* são aqueles que ligam duas viagens cujo tempo de espera entre elas é grande suficiente para que o veículo retorne à garagem, permanecendo lá um intervalo mínimo de tempo antes de executar a próxima viagem. Este "retorno temporário"à garagem é importante do ponto de vista prático pois permite realizar o abastecimento e serviços rápidos no veículo, além de criar a oportunidade para a troca da tripulação, pausa para descanso e/ou alimentação, execução de outras tarefas, ou mesmo a sua dispensa por um determinado período de tempo. A omissão destes arcos na formulação do problema leva à geração de programações de baixa qualidade.

b) *arcos curtos:* são aqueles que ligam viagens cujo tempo de espera entre elas não permite o retorno temporário do veículo à garagem.

Tomando como referência a rede genérica apresentada na figura (3.1), define-se o conjunto de arcos longos $A^l \subset A$ como sendo as ligações entre as viagens que satisfazem a seguinte relação:

$$(i,j) \in A^l \Leftrightarrow d_j - a_i \geq t_{is} + \textit{tempo mínimo de garagem} + t_{rj} \tag{4.1}$$

onde o *tempo mínimo de garagem* é um parâmetro definido pelo usuário. O conjunto dos arcos curtos A^c é dado por $A^c = A - A^l$. Embora a maioria dos arcos na rede que representa o problema seja constituída de arcos longos, apenas uma pequena parcela destes arcos toma parte da solução ótima. Portanto, este conjunto de arcos é o mais propício a ser reduzido. A tabela (4.1) mostra a participação dos arcos longos na composição da rede e da solução ótima para alguns casos reais.

Tabela 4.1: Total de arcos curtos e longos na representação de diferentes problemas.

Viagens	Curtos A^c	Longos A^l	Curtos na Sol.	Longos na Sol.	Veículos
208	4.023	14.686	207	0	14
388	11.790	54.507	388	0	23
887	51.757	299.797	886	0	47
1.134	83.022	491.322	1.130	3	59
1.518	147.610	871.828	1.514	3	84

Dentre as três técnicas de redução da rede apresentas por Freling et al. (1995), foram exploradas neste trabalho as duas mais genéricas, descartando aquela que depende da estrutura do problema. Estas técnicas, sitadas na seção (3.7) e delhadas abaixo, ainda não haviam sido devidamente exploradas, principalmente a de geração dos arcos longos.

4.2 A Técnica de Eliminação dos Arcos Longos

A técnica proposta por Bokinge e Hasselström (1980) e adaptada por Freling et al. (1995) é de substituir os arcos longos por retornos à garagem através de arcos já existentes na rede. Para tanto, deve-se executar um processo que inicialmente substitui os nós r e s por uma seqüência de $2n$ nós garagem, ou seja, para cada viagem $i \in V$ cria-se um nó de partida da garagem r_i e outro de chegada à garagem s_i. Estes $2n$ nós garagem são dispostos em ordem crescente, de acordo com seus

horários de ocorrência, e são ligados seqüencialmente por arcos com custo zero e capacidade n. Existe ainda um arco de retorno ligando o último nó garagem ao primeiro nó garagem, com capacidade n e custo igual ao Custo Fixo, garantindo a circulação do fluxo e contabilizando o número mínimo necessário de veículos. Cada viagem i tem o seu nó de partida da garagem ligado ao seu nó de início por um arco com capacidade um e custo c_{ri}, e o seu nó de final de viagem é ligado ao nó de retorno à garagem com capacidade um e custo c_{is}, custos estes dados pela expressão (3.1), sem o Custo Fixo. O tamanho desta rede é reduzido agrupando-se os nós garagem da seguinte maneira: agrupar os nós da lista ordenada de garagens, iniciando um novo grupo sempre que um nó de retorno à garagem for encontrado, visto que neste grupo já existe um nó de partida. Cada grupo é substituído por um novo nó, o qual receberá e enviará os arcos incidentes e provenientes dos nós do grupo. Este processo garante que um veículo que sai de um nó de partida da garagem tenha retornado previamente à garagem. A figura (4.1) mostra a aplicação desta técnica à rede da figura (3.3), assumindo que o tempo mínimo de garagem seja de 30 minutos e que as viagens mortas entre o terminal A e a garagem duram 5 minutos.

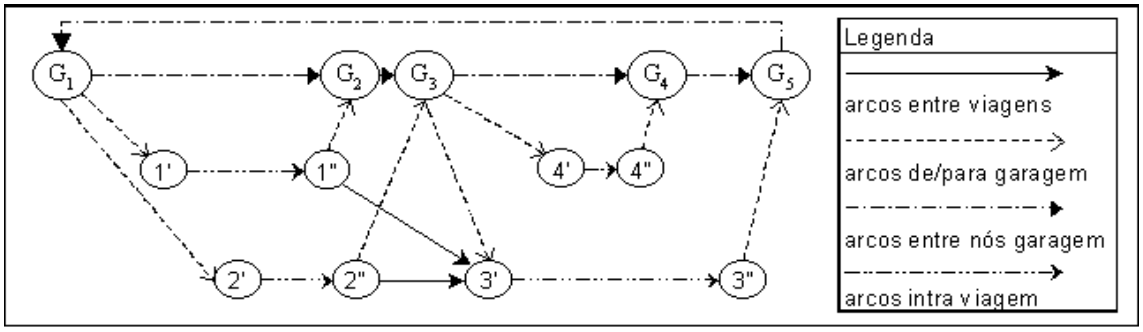

Figura 4.1: Representação do PPV como Problema de Designação.

Assim, é possível eliminar os arcos $(1'', 4')$ e $(2'', 4')$, que neste caso são classificados como arcos longos. No exemplo não houve redução do número total de arcos devido à simplicidade da rede, porém os testes com casos reais mostraram uma diminuição na densidade da rede, como pode ser visto nos resultados apresentados no capítulo (5).

4.3 Análise da Técnica de Eliminação de Arcos

Embora a eliminação dos arcos longos aplicada ao problema de fluxo com custo mínimo reduza consideravelmente a rede resultante, duas limitações associadas a esta técnica foram verificadas neste trabalho.

A primeira limitação diz respeito às dificuldades de se introduzir restrições adicionais ao modelo. Uma restrição encontrada com freqüência em situações práticas exige que o veículo, caso retorne temporariamente à garagem, deva permanecer no local pelo menos durante o "tempo mínimo de garagem". A técnica de eliminação de arcos longos não comporta tal restrição, pois o agrupamento dos nós garagem leva em conta a ordem de ocorrência destes nós e não o tempo de garagem dos veículos. Com isso, não é possível detectar o tempo que o veículo permaneceu na garagem antes de retornar à operação. Esta limitação pode ser contornada criando-se uma garagem temporária a cada 30 minutos (Bokinge e Hasselström 1980), porém com esta alternativa o tempo mínimo de espera na garagem pode variar entre 30 e 59 minutos, dependendo da situação, o que não corresponde com exatidão ao problema real.

As restrições de *preferência de ligação* visam priorizar o sequenciamento de viagens pertencentes a duas ou mais rotas pré-determinadas. Nestes casos o sequenciamento de viagens de rotas com preferência de ligação têm seus custos reduzidos de um dado valor e não seguem a função de custo definida em (3.1), como é o caso dos outros arcos. Esta é uma restrição encontrada com freqüência e não pode ser considerada quando aplicada a técnica de eliminação de arcos longos, pois ao substituir um arco por arcos já existentes na rede, perde-se o controle do veículo envolvido na execução de duas viagens entremeadas com um retorno temporário à garagem, impossibilitando assim a aplicação do redutor do custo da ligação.

A segunda limitação do método de eliminação dos arcos longos é o tempo total de processamento, que permanece elevado quando comparado com a segunda técnica de redução.

4.4 Técnica de Geração de Arcos

Esta técnica é uma versão em redes da geração de colunas da programação linear, que tem sido largamente utilizada na resolução de problemas de grande porte. A geração de colunas permite solucionar o problema de otimização sem levar em conta todas as colunas da matriz. Assim, vários problemas de programação de veículos e de suas tripulações foram resolvidos através desta técnica, incluindo a programação de ônibus, trens, e aeronaves (Crainic e Rousseau 1987, Desrochers e Soumis 1989, Fores 1996 e Fores et al. 1999). Trabalhos recentes como o de Löbel (1998) e Barnhart et al. (1998) propõem uma metodologia de *branch-and-price*, que é um processo de *branch-and-bound* combinado com geração de colunas para resolver problemas de programação inteira mista de grande porte. Löbel (1998) usa uma metodologia similar para resolver o Problema de Programação de Veículos com Várias Garagens.

Como a técnica descrita neste seção é uma adaptação da geração de colunas proveniente da teoria de programação linear, é apresentada a seguir a idéia do método que deu origem à técnica de geração de arcos.

4.4.1 Método de Geração de Colunas

O método de geração de colunas é uma estratégia que auxilia o método Simplex a resolver PPLs de grande porte, no qual a matriz A possui o número de colunas muito maior do que o número de linhas. Esta situação surge em diversos problemas clássicos, tais como o problema de corte de papel (*cutting-stock problem*) e o problema de programação de tarefas, veículos e máquinas (genericamente denominados *scheduling problem*). A geração de colunas é utilizada principalmente como uma subrotina do *Princípio de Decomposição de Dantzig-Wolfe*, que pode ser aplicado quando a matriz for particionável em dois conjuntos: um problema principal e outros problemas menores (Dantzig e Wolfe 1960). Em vez de trabalhar com a matriz A, parte-se de uma submatriz A_s com um número de colunas bem inferior ao da matriz completa, porém que contenha pelo menos uma solução viável. Uma vez resolvido o *subproblema* definido sobre o domínio A_s, utilizam-se as informações da solução ótima corrente para encontrar uma coluna que ainda não foi considerada, e que não

está de acordo com as condições de otimalidade do Simplex. Esta nova coluna é encontrada resolvendo-se um segundo PPL. Caso exista uma coluna candidata, ela é incluído no subproblema, caso contrário a solução corrente é a solução ótima do problema original.

A utilização do método de geração de colunas está intrinsicamente associada ao Simplex Revisado, o qual é descrito sucintamente a seguir.

4.4.1.1 O Método Simplex Revisado

No método Simplex, cada iteração é iniciada escolhendo-se a variável a entrar na base, seguida pela escolha da variável que deverá sair da base, e finalizada com a atualização da solução básica corrente. Como o Método de Geração de Colunas está associado ao esforço de encontrar uma variável não-básica candidata a entrar na base, será discutida a seguir apenas a etapa que trata especificamente deste ponto.

Considere o seguinte (PPL) na forma normal, ou seja com restrições de igualdade:

$$\begin{array}{lrcl} \text{Max} & Z & = & c^T x \\ \text{sujeito a} & Ax & = & b \\ & x & \geq & 0 \end{array} \tag{4.2}$$

Cada solução ótima x^* do problema pode ser particionada em $x_1, x_2, ..., x_{m+n}$, onde as m primeiras variáveis são ditas *variáveis básicas*, pois suas respectivas colunas constituem uma base B da matriz A. Tais variáveis serão denotadas por x_B. As n variáveis restantes são ditas *variáveis não básicas*, denotadas por x_N e cujo conjunto das respectivas colunas será referenciado por A_N. Este fato induz a uma partição de x^* em x_B e x_N, uma partição de A em A_B e A_N, assim como de c em c_B e c_N. Desta forma, a equação $Ax = b$ pode ser expressa como:

$$Bx_B + A_N x_N = b \tag{4.3}$$

Multiplicando-se a equação acima por B^{-1} pela esquerda, tem-se:

$$x_B = B^{-1}b - B^{-1}A_N x_N \tag{4.4}$$

Substituindo a expressão acima no equação (4.3) obtém-se:

$$Z = c_B B^{-1} b + (c_N - c_B B^{-1} A_N) x_N \tag{4.5}$$

No Simplex padrão, o vetor $c_N - c_B B^{-1} A_N$ é calculado juntamente com a atualização do sistema linear. Já no Simplex Revisado esta informação pode ser obtida individualmente para uma determinada variável candidata. O cálculo é realizado em duas etapas: inicialmente o vetor $y = c_B B^{-1}$ é encontrado resolvendo-se o sistema $yB = c_B$. Posteriormente calcula-se o termo $c_N - yA_N$. Assim, considerando x_j uma variável não-base cuja coluna correpondente em A_N seja a, então seus coeficientes atualizados na função objetivo são dados por $c_j - ya$. Logo, para que esta variável seja candidata a entrar na base ela deve satisfazer à inequação $c_j > ya$.

Visto que a componente $c_N - yA_N$ pode ser calculada individualmente, mais de uma variável candidata pode ser encontrada calculando-se o vetor $c_N - yA_N$ ou parte dele. As estratégias que se utilizam desta característica do Simplex Revisado são genericamente conhecidas como *pricing*.

Em alguns casos determina-se a variável a entrar na base por meio de um segundo problema de otimização. Esta técnica, chamada de *geração de colunas*, é parte do Princípio de Decomposição de Dantzig-Wolfe.

4.4.1.2 Geração de Colunas

Considere agora o seguinte PPL:

$$\begin{aligned} \text{Min} \quad & Z = \textstyle\sum_{j=1}^{n} c_j x_j \\ \text{sujeito a} \quad & \textstyle\sum_{j=1}^{n} p_j x_j = b \\ & x_j \geq 0, \ j = 1, ..., n \end{aligned} \tag{4.6}$$

onde p_j e b são vetores com m componentes, $m < n$. Seja x_B uma solução básica inicial associada à matriz básica B e aos coeficientes de custo c_B. Então, as variáveis duais associadas a esta base $\pi = c_B B^{-1}$ estão sempre disponíveis.

Para melhorar a solução básica é necessário avaliar todas as colunas correspondentes às variáveis não-básicas, computando seus coeficientes de custo relativo dados por:

$$\bar{c}_j = c_j - B^{-1} \tag{4.7}$$

Se

$$\min_j \bar{c}_j = c_s < 0 \tag{4.8}$$

então, desconsiderando degenerecência, a solução corrente pode ser melhorada ao introduzir x_s na base.

Se o problema tiver um número de colunas muito grande, então encontrar o valor de c_s em (4.8) pelo cálculo e comparação de cada $\bar{c}_j$ pode implicar um grande esforço computacional. Entretanto, nos problemas de grande porte, o conjunto de colunas tem uma estrutura bem definida, resultante das particularidades da situação real. Em geral, pode-se assumir que as colunas p_j fazem parte de um conjunto S, que é o conjunto de vetores que satisfazem algum sistema de equações ou inequações. Assim, as colunas a entrarem na base podem ser escolhidas resolvendo-se o subproblema

$$\begin{array}{ll} \text{Min} & c(p_j) - \pi p_j \\ \text{sujeito a} & p_j \in S \end{array} \tag{4.9}$$

onde $c(p_j) \equiv c_j$ é uma possível função de p_j. Dependendo da estrutura de S e da forma do vetor c, varias técnicas podem ser usadas para resolver este subproblema. Esta técnica é dita "geração de colunas" pois, resolvendo-se o subproblema, apenas um pequeno subconjunto de colunas em S é examinado e gerado quando necessário.

Os problemas de programação linear podem ser grandes não só em número de variáveis, mas também em número de restrições. Nestes casos pode-se transformar o problema em um problema equivalente que contém um número de variáveis muito maior do que o original. O método de geração de colunas é então aplicado ao problema equivalente. Esta é a idéia básica do Princípio de Decomposição de Dantzig-Wolfe.

4.4.2 O Algoritmo de Geração de Arcos

O problema de fluxo em redes é um caso particular de um problema de programação linear no qual a matriz associada à rede, isto é, a *matriz de incidência*, apresenta em suas linhas as informações dos nós e em suas colunas as informações dos arcos. Assim, ao gerar uma nova coluna desta matriz, um novo arco está sendo gerado na rede. As colunas da matriz apresentam as informações dos arcos, por isso elas são

compostas por apenas dois elementos diferentes de zero, os quais se referem aos nós nas extremidades do arco. Esta característica da matriz torna a técnica de geração de arcos um processo mais simples do que na geração de colunas, visto que o cálculo do custo relativo dos arcos não envolve a matriz inversa da base, mas simplesmente os valores duais da solução corrente. Uma vez que os valores duais fazem parte da solução ótima, nenhum esforço computacional extra é necessário para se conhecer os arcos com capacidade de melhorar a solução corrente. Nesta adaptação o termo "geração de arcos"deve ser considerado no sentido de Schrijver (1989), ou seja, como uma técnica de inclusão de novos arcos ao problema, baseada na avalição de seus custos relativos (*pricing based technique*). O conjunto dos arcos que não fazem parte do processo de otimização são armazenados e pesquisados com a finalidade de gerar um novo domínio para o problema.

A estratégia de geração de arcos consiste em resolver uma seqüência de problemas, iniciando com um domínio composto por um pequeno conjunto de arcos com uma grande possibilidade de participar da solução. A partir dos valores duais fornecidos pela solução do problema corrente é feita uma busca na lista de arcos fora do domínio, identificando aqueles que têm capacidade de melhorar a solução (esta etapa não é utilizado algoritmo de caminho mínimo, como ocorre na geração de colunas). Se forem encontrados arcos fora do domínio que possam melhorar a solução corrente, então esses arcos serão incorporados à rede e o problema é resolvido novamente. Este processo é repetido até que nenhum arco fora do domínio possa melhorar a solução. Quando isto ocorrer o problema estará resolvido e a solução corrente será a ótima. A representação esquemática abaixo sintetiza a técnica de geração de arcos.

Algoritmo de Geração de Arcos

Inicialização: Dada uma rede $G = (N, A)$, seja S e L conjuntos tais que $S \cup L = A$.

Passo 1: Resolver o problema de Pseudo-Designação definido em (3.10) - (3.15) tendo como domínio o subconjunto S.

Passo 2: Para cada arco $(i, j) \in L$ faça:

Calcule $\bar{c}_{ij} = c_{ij} - \pi_i - \pi_j$

Se $\bar{c}_{ij} < 0$

Então incluir o arco (i, j) a S e retirá-lo de L.

Passo 3: Faça

Se algum arco foi incluído em S durante o passo Passo 2

Então retornar ao Passo 1,

Senão finalizar o processo. A solução corrente é ótima.

A partir dos dados de entrada, o PPV é formulado através do modelo de pseudo-designação, cuja rede correspondente é apresentada na figura (3.5). O conjunto S é inicializado com o conjunto A^c dos arcos curtos. Isto significa que são consideradas apenas as ligações entre viagens que não permitem o retorno temporário do veículo à garagem. O conjunto L é inicializado com os arcos longos, ou seja $L = A^l$.

Esta técnica inclui novos arcos ao domínio do problema, buscando-os no conjunto contendo aqueles arcos que não fazem parte do problema. Na geração de colunas segundo Dantzig-Wolfe, esta busca é substituída pela execução sucessiva do algoritmo de caminho mínimo.

4.5 Análise do Método de Geração de Arcos

Com a técnica de geração de arcos é possível reduzir substancialmente o tempo de processamento, chegando a consumir apenas 15% do tempo requerido pela técnica de eliminação dos arcos longos. Logo, torna-se possível resolver eficientemente problemas de programação de veículos de grande porte.

Embora o domínio do problema seja composto por uma pequena parcela dos arcos da rede, todos os arcos são pesquisados e suas características analisadas tendo em mente os critérios de otimalidade. Portanto, nenhuma informação da rede é perdida durante o processo, tornando possível incorporar ao modelo práticas operacionais freqüentes no sistema de transporte público, como o retorno temporário à garagem e a preferência de ligação entre determinadas linhas. A forma como estas restrições foram incluídas no modelo serão descritas em um capítulo posterior.

4.6 Conclusões

As duas técnicas apresentadas mostram-se capazes de reduzir o número de arcos na rede. Entretanto, a Técnica de Geração de Arcos conta com uma característica muito desejável no estudo de casos reais. Ela permite a inclusão de restrições adicionais provenientes da prática operacional.

Um fator determinante na escolha da técnica a ser adotada diz respeito ao seu desempenho computacional. Os testes de desempenho realizados com as técnicas são apresentados a seguir.

Capítulo 5

Desempenho dos Métodos de Redução

5.1 Introdução

Para verificar a eficiência dos métodos quando aplicados a casos reais e de grande porte, foram gerados problemas de diferentes tamanhos, a partir de dados de empresas que operam na cidade de *Reading* - Reino Unido e de Sorocaba no Brasil. Os dados de Sorocaba se referem a casos reais de empresas que operam naquela localidade, cujo problema de programação da tripulação já foi abordado por Wren e Gualda (1999). Neste capítulo são apresentados os testes computacionas realizados com os seguintes objetivos:

a) verificar a capacidade de cada técnica estudada em reduzir o tamanho de suas respectivas redes que representam o problema, e

b) comparar o tempo de processamento requerido por cada método com sua respectiva técnica, para obter a solução ótima.

Os métodos podem ser avaliados em função de suas capacidades de redução da rede que representa o problema, ou em relação ao tempo de processamento para a obtenção da solução ótima. Como a capacidade de armazenamento de dados tem crescido consideravelmente nas últimas gerações de computadores, torna-se decisivo o fator "tempo de processamento" requerido por cada método. Para uma análise completa de eficiência, são apresentados a seguir os dois aspectos dos diferentes

métodos.

Os dados utilizados nos testes comparativos, assim como o acesso ao sistema *BOOST* foram disponibilizados pelo Grupo de Programação de Veículos e Tripulação da Universidade de *Leeds* durante estágio no Reino Unido, realizado no decorrer deste trabalho de pesquisa.

5.2 Eficiência na Redução da Rede

As duas técnicas abordadas conseguem reduzir sensivelmente o tamanho da rede. Porém, a técnica de eliminação dos arcos longos produz redes com densidades mais baixas do que a técnica de geração de arcos, como pode ser observado no gráfico da figura (5.1). A densidade de uma rede com n nós e m arcos é definida por m/n^2m, e tem como finalidade criar um índice para comparar redes distintas. Uma rede com $m = O(n^2)$, ou seja, na qual m é da ordem de n^2, é considerada uma rede *densa*. Se $m = O(n)$, então a rede é considerada *esparsa*. No caso, as densidades apresentam o mesmo padrão de comportamento diferindo apenas por uma constante. Ao aplicar

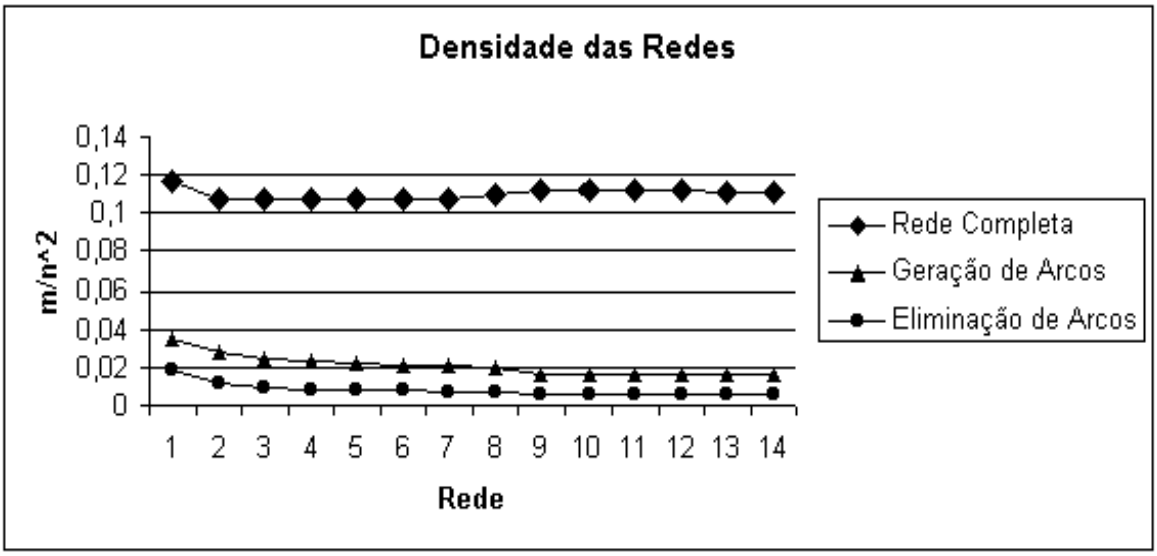

Figura 5.1: Densidade da rede completa, aplicando Eliminação dos Arcos Longos e Geração de Arcos.

a técnica de eliminação dos arcos longos verificou-se um aumento médio de 50% no número de nós, no entanto houve uma redução de até 89% dos arcos longos e uma redução significativa na densidade da rede, passando do intervalo [10,05; 11,65]

para o intervalo [0,54; 1,88]. Como conseqüência da redução no tamanho da rede, ocorre também uma redução no tempo de processamento, em relação ao tempo de processamento da rede completa.

Através da técnica de geração de arcos longos foi possível reduzir o tamanho da rede e, conseqüentemente, o tempo de processamento. Neste caso, a taxa de redução da rede variou entre 69,47% e 85,20%, e a densidade caiu para o intervalo [3,56; 1,65]. Embora a técnica anterior tenha obtido melhores resultados quanto à taxa de redução de arcos e à densidade das redes, a geração de arcos mostrou-se mais eficiente no tempo de processamento e na flexibilidade do método, permitindo introduzir restrições adicionais, que não foram possíveis de incluir quando aplicada a técnica anterior.

5.3 Tempo de Processamento

A técnica de geração de arcos associada ao modelo de pseudo-designação não apresenta a melhor taxa de redução de arcos, no entanto o tempo de processamento requerido por esta metodologia chega a 14,79% do tempo requerido pela técnica de eliminação dos arcos longos. Para o maior caso em que a rede completa foi resolvida, a técnica de geração de arcos levou a uma redução de 84,98% do tempo de processamento enquanto a técnica de eliminação de arcos reduziu em apenas 35,28% em relação ao tempo de processamento da rede completa.

Os onze problemas relativos à cidade de *Reading* foram resolvidos considerando a rede completa e aplicando os dois métodos de redução da rede apresentados neste trabalho. A figura (5.2) mostra os tempos requeridos por cada método para resolver os respectivos problemas. Os problemas foram dispostos em ordem crescente de tamanho para se estabelecer uma relação entre o desempenho dos métodos com o crescimento do problema.

Pode-se verificar que o tempo de processamento da rede completa cresce segundo uma curva exponencial, o mesmo acontecendo, embora em um grau menos acentuado, com o método de eliminação de arcos. Tal característica mostra que embora haja uma redução no tamanho da rede, a eliminação dos arcos longos não teve a capacidade de retirar a característica *NP-Hard* dos problemas. Por outro lado, o tempo de

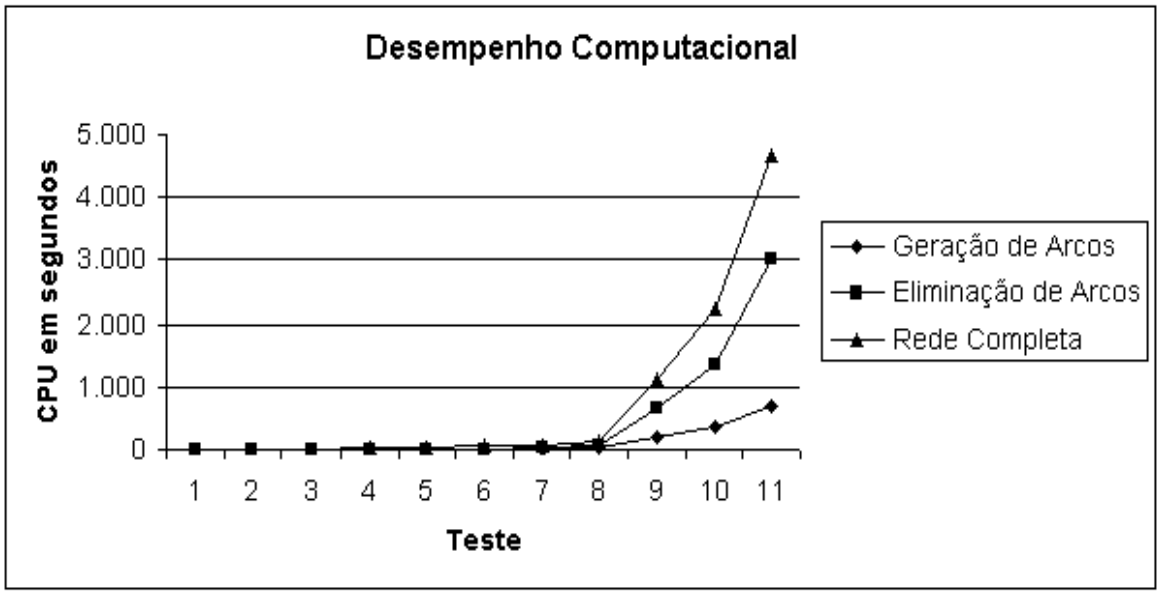

Figura 5.2: Tempo de CPU requerido pelos métodos de redução de arcos e computando a rede completa.

processamento referente à técnica de geração de arcos mostra que esta abordagem transforma um problema *NP-Hard* em um problema cujo tempo de processamento cresce linearmente em relação ao tamanho da rede.

5.4 Resolução de Problemas de Grande Porte

Para testar a eficiência do método proposto na resolução de problemas de grande porte, foram selecionadas as situações cujas redes apresentam as maiores dimensões dentre os casos estudados. Estes problemas foram resolvidos com o algoritmo descrito da seção (6.2) do capítulo (6).

A tabela (5.1) contém um conjunto de experimentos que mostra como o método se comporta quando o número p varia entre 10 e 25, para um problema com 2.613 viagens, requerendo 107 veículos. A rede que representa este problema como uma pseudo-designação é composta por 5.228 nós e 3.128.723 arcos. Em todos os casos o método atingiu a solução ótima.

Tabela 5.1: *Trade-off* do Algoritmo de Geração de Arcos Melhorado.

(p)	Arcos longos incluídos	Total de arcos	Subproblemas	CPU (min)
10	138.020	444.231	40	34,20
15	155.472	461.683	28	27,00
20	165.467	471.678	20	23,35
25	191.754	497.965	18	22,57

5.5 Conclusão

Os testes computacionais, realizados com dados reais, mostram que a técnica de Eliminação dos Arcos é mais eficiente na redução da rede, do que a de Geração de Arcos. Por outro lado, a Geração de Arcos conta com:

i) o menor tempo de processamento, e

ii) a capacidade de considerar restrições adicionais inerentes à prática operacional.

Logo, fica clara a predominância da Técnica de Geração de Arcos, sendo portanto a mais adequada para o desenvolvimento de uma metodologia capaz de abordar casos reais com diferentes características operacionais e de grande porte. O próximo capítulo mostra como representar o PPV para que a técnica de geração de arcos seja associada ao algoritmo *Out-of-Kilter*.

Capítulo 6

O Método Proposto para o PPV

6.1 Introdução

Este trabalho explora a utilização da técnica geração de arcos, associado-a ao algoritmo de fluxo em redes *Out-of-Kilter* para resolver o PPV. Para aplicar o algoritmo *Out-of-Kilter*, a rede dever ser *fortemente conectada*. A rede que apresenta tal característica é a de fluxo com custo mínimo apresentada na figura (3.3), que pode ser vista com uma adaptação da rede de pseudo-designação.

Este capítulo apresenta a estratégia utilizada para transformar o problema de pseudo-designação em um problema de circulação, uma vez que o algoritmo *out-of-kilter* exige que a rede seja conectada. Também são apresentadas as flexibilizações acrescidas ao modelo que o tornam mais abrangente e capaz de resolver diferentes problemas da realidade brasileira.

6.2 Geração de Arcos e Algoritmo *Out-of-Kilter*

Para que o algoritmo *out-of-kilter* seja aplicado a uma determinada rede, é necessário que ela seja uma rede conectada. Esta característica pode ser garantida acrescentando-se um determinado arco para cada viagem, operação esta que transforma o problema de pseudo-designação em um problema de circulação.

6.2.1 O Problema de Pseudo-Designação como Circulação

Dada a rede $G^{pd} = (N_1, N_2, V)$ definida na seção (3.6), considere a inclusão de arcos auxiliares ligando o nó início de cada viagem i' ao seu nó de término i''. Ao aplicar esta idéia à rede da figura (3.5), obtém-se a rede na figura (6.1). É fácil verificar que esta rede nada mais é do que aquela na figura (3.3), porém com os nós dispostos de forma distinta.

A esta rede (6.1 ou 3.3) pode ser aplicada a técnica de geração de arcos, possibilitando resolver de forma eficiente o PPV. Pode-se também considerar algumas adaptações, que tornam possível gerar programações que utilizam um número veículos menor do que a frota mínima. Tais programações são alcançadas com o não cumprimento de algumas viagens planejadas. Desta forma é possível explorar diferentes situações operacionais, ou mesmo realizar o planejamento da tabela de horários, tomando como base dados inerentes às viagens, tais como a taxa de ocupação do veículo na viagem, intervalo de tempo até a próxima partida, entre outros.

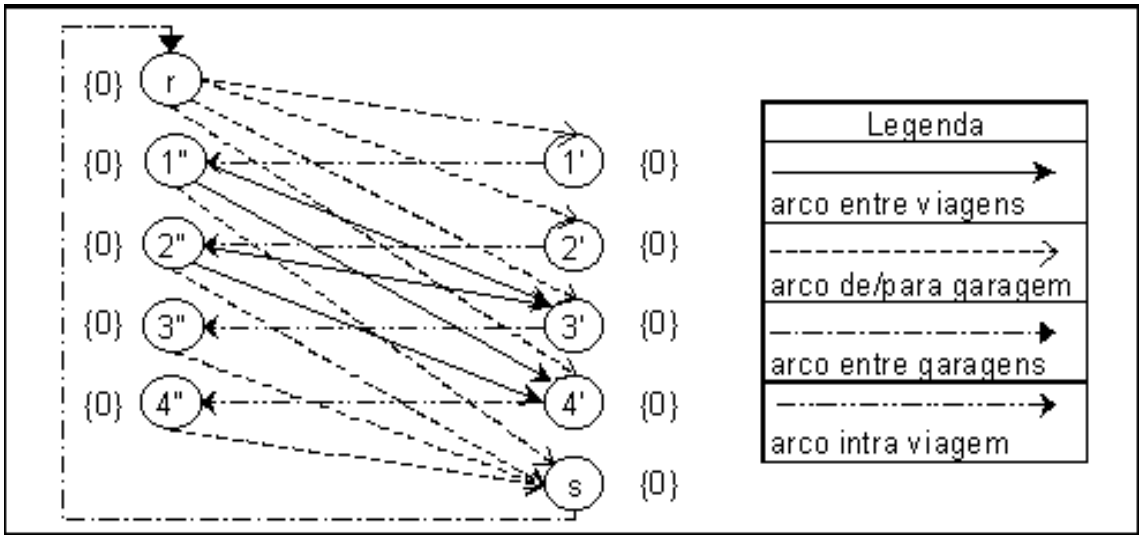

Figura 6.1: Rede G^{pd} transformada em uma rede de Circulação.

O modelo matemático para este problema é o mesmo do problema de fluxo com custo mínimo apresentado em (3.3) - (3.5).

6.2.2 O Algoritmo de Geração de Arcos para o PPV

Para resolver os problemas abordados, foi utilizada a técnica de geração de arcos (seção 4.4), combinada com o algoritmo *Out-of-Kilter* (seção 2.6). O algoritmo é descrito como segue.

Dado um conjunto de viagens V, montar a rede de circulação equivalente à rede (6.1), conforme descrito acima. Seja S o conjunto dos arcos curtos e L o conjunto dos arcos longos, então define-se:

ArcGen - Algoritmo de Geração de Arcos combinado com o Out-of-Kilter

Inicialização: Dada uma rede conectada $G = (N, A)$, seja S o conjunto dos arcos curtos e L o conjunto dos arcos longos.

Passo 1: Utilizando o algoritmo *Out-of-Kilter* resolver o problema de Circulação definido por (3.3) - (3.5) para a rede reduzida $G_R = (N, S)$.

Passo 2: Para cada arco $(i, j) \in L$ faça:

Calcule $\bar{c}_{ij} = c_{ij} - \pi_i - \pi_j$

Se $\bar{c}_{ij} < 0$

Então incluir o arco (i, j) a S e retirá-lo de L.

Passo 3: Faça

Se algum arco foi incluído em S durante o passo Passo 2

Então retornar ao Passo 1,

Senão finalizar o processo. A solução corrente é ótima.

Este algoritmo foi utilizado como base para o desenvolvimento dos estudos subseqüentes. As flexibilizações apresentadas na seção seguinte foram implementadas a partir deste algoritmo.

Para abordar problemas de grande porte, foi feita uma adaptação que reduz ainda mais o conjunto de arcos processados. Entretanto, um número maior de problemas deve ser resolvido em relação ao algoritmo acima. Tal versão do algoritmo é descrita a seguir.

6.2.3 A Geração de Arcos para Problemas de Grande Porte

Para problemas grandes, mesmo aplicando a técnica de geração de arcos, a rede resultante conta com um elevado número de arcos. Para minimizar este problema, a técnica foi alterada, limitando-se a inclusão de arcos longos em cada iteração. Neste caso é permitida a inclusão de no máximo p arcos longos por viagem ao subproblema a ser reotimizado.

Considerando o PPV dado pela rede conectada $G = (N, A)$, seja p um limitante superior para o número de arcos a ser incluído na rede para cada viagem. Assim, a adaptação do método de geração de arcos torna-se:

Algoritmo ArcGen para Redes de Grande Porte

Inicialização: Dada $G = (N, A)$ uma rede conectada, seja S o conjunto dos arcos curtos e L o conjunto dos arcos longos.

Passo 1: Resolva o problema de circulação para a rede reduzida $G_R = (N, S)$. A solução fornece o vetor dual de preço no nós π.

Passo 2: Para cada arco $(i, j) \in L$ calcule $\bar{c}_{ij} = c_{ij} - \pi_i - \pi_j$.

Passo 3: Para cada viagem $i \in V$, escolha os p arcos (i, j) com custos mais negativos, caso existam. Incluir estes arcos no subconjunto S e retirá-los de L.

Passo 4: Faça

Se algum arco foi incluído em S durante o passo Passo 3

Então retornar ao Passo 1,

Senão finalizar o processo. A solução corrente é ótima.

Esta variante da técnica de geração de arcos leva a situações com diferentes relações de custo/benefício. Ou seja, a série de subproblemas resolvidos no Passo 1 e o tempo de CPU diminuem à medida que o número p aumenta. Por outro lado, o número de arcos na rede final aumenta com p, exigindo da máquina maior capacidade de armazenamento em tempo de execução (memória RAM). Portanto, esta relação de custo/benefício pode ser enunciada assim:

O aumento no número p leva a uma diminuição no tempo de CPU, mas requer maior capacidade de processamento da máquina, e vice-versa.

6.2.4 A Função de Custo do Algoritmo *ArcGen*

Uma atenção especial deve ser dada à função de custo adotada no modelo. Isso se deve ao fato de que, na grande maioria dos casos, a frota já foi adquirida, restando como objetivo minimizar a parcelo do custo total envolvido na operação, e que é de fato otimizável. Analisando um sistema de transporte público, verifica-se que otimizar sua operação corresponde a minimizar os períodos de ociosidade do veículo e da tripulação. Assim sendo, deve-se estabelecer os diferentes custos de ociosidade do veículo e incorporar tais custos à função objetivo a ser minizada.

As funções de custo (3.1) e (3.2) apresentadas no capítulo (3) levam em conta os aspectos citados acima, sendo que na primeira função os custos relativos à tripulação não estão explitados como o estão na segunda função.

Neste trabalho foi adotada a expressão (3.1) onde o *Custo Fixo* está associado ao custo de cada veículo. A constante K_1 representa o custo por minuto que o veículo roda ociosamente, e K_2 é o custo por minuto do veículo parado no terminal.

6.3 Flexibilizações do Modelo

O modelo foi adaptado para atender às principais características operacionais peculiares ao sistema de transporte público brasileiro, tornando possível utilizá-lo na resolução de um grande número de casos reais. Abaixo são descritas as flexibilizações introduzidas no modelo.

6.3.1 O Problema do Retorno à Garagem

Para justificar o retorno de um veículo à garagem no intervalo entre duas viagens, é necessário que o mesmo permaneça lá por um período mínimo de tempo definido previamente pelo operador, denominado *tempo mínimo de garagem*. Durante este intervalo de tempo em que o veículo permanece na garagem, o custo operacional é tomado como constante, independente de sua duração. Esta consideração implica uma adaptação da função do custo dos arcos longos. Isto significa que os arcos longos têm seu custo dado por:

$$(i,j) \in A^l \Rightarrow c_{ij} = K_1(t_{is} + t_{rj}) + K_2(\textit{tempo mínimo de garagem}) \qquad (6.1)$$

Segundo a expressão (6.1), o custo atribuído aos arcos longos é composto por duas parcelas. Na primeira é considerado o custo de deslocamento terminal-garagem-terminal. Na segunda parcela, o custo de permanência na garagem é tido como constante, independente da sua duração. Esta estrutura de custo é adotada na grande maioria das situações práticas por estar de acordo com as premissas de minimização de custos (Dell'Amico et al. 1993, Kwan e Rahin 1999). Ao adotar esta estrutura de custo para os arcos longos, é possível controlar o retorno dos veículos à garagem de acordo com a prática da empresa. Aumentando-se o parâmetro *tempo mínimo de garagem* restringe-se o número de retornos temporários à garagem.

6.3.2 Problema de Preferência de Ligações

Em alguns casos o operador do sistema sabe de antemão que certas viagens que chegam num terminal devem ser ligadas preferencialmente com viagens de determinadas linhas que partem deste terminal. Isto se deve por razões operacionais, tais como a previsão de atrasos em rotas sobrecarregadas, fluxo de passageiros em determinado sentido, definindo um padrão de demanda que envolve mais do que uma rota, condições para a programação da tripulação, etc. Nestes casos o operador pode admitir uma certa prioridade na ligação entre determinadas rotas.

Esta situação denominada *preferência de ligação* é definida por dois conjuntos de rotas e um *fator de redução* no custo da ligação. As viagens do conjunto das *rotas de chegada* têm preferência para serem ligadas às viagens do conjunto das *rotas de partida*, tendo seu custo reduzido pelo fator de redução das ligações ocorridas entre tais rotas. Uma rota pode ter prioridade de ligação em relação a mais do que uma outra rota, o que é denominado preferência de ligações múltiplas.

Este tipo de restrição também altera a estrutura da função de custo dos arcos e é compatível com aplicação da técnica de geração de arcos. Ao definir o custo de um arco, verifica-se se as linhas às quais pertencem as viagens envolvidas apresentam preferência de ligação, aplicando-se então o fator de redução do custo da ligação.

6.3.3 A Redução do Número de Viagens

Para considerar a possibilidade da não realização de uma viagem, é necessário acrescentar mais um arco para cada viagem planejada i. Este arco, dito *arco de auto-*

atribuição, liga o nó final da viagem i'' ao seu início i', e pode ser interpretado como um veículo voltando no tempo e retroalimentando a viagem. Para melhor visualização, os arcos de auto-atribuição foram incluídos na rede (3.3), resultando na rede da figura (6.2). Fisicamente, prover uma viagem com um veículo que acaba

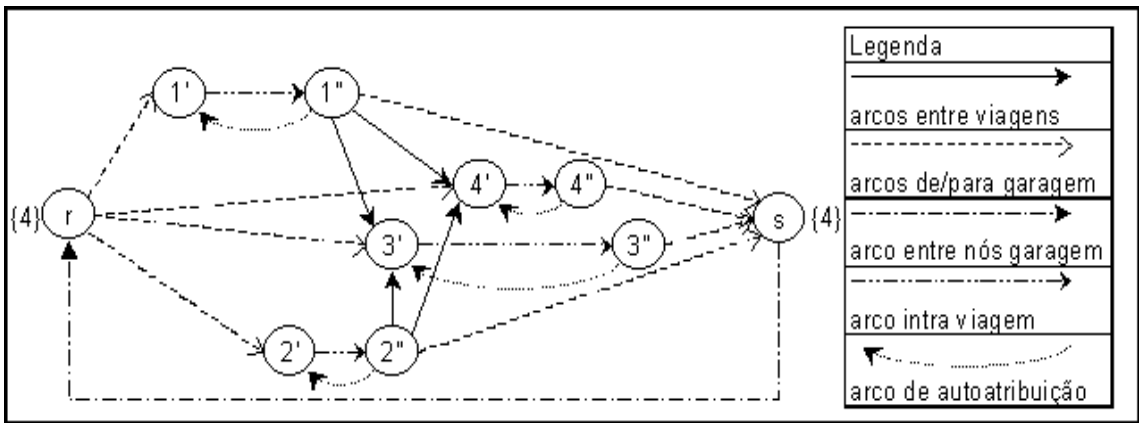

Figura 6.2: Rede com arcos de auto-atribuição.

de realizá-la é um absurdo, mas esta idéia é utilizada para possibilitar a omissão de viagens na programação final. Quando um arco de auto-atribuição faz parte da solução, a viagem associada não entrará na programação gerada, e portanto não será realizada.

O controle da participação destes arcos na solução está associado ao *custo de omissão* c_{oi} associado a cada uma das viagens. Quanto maior o custo de omissão c_{oi} maior será a chance da viagem i participar da programação e, conseqüentemente, de ser executada. Reciprocamente, quando c_{oi} for suficientemente pequeno, a viagem i não fará parte da programação. No estudo de caso relativo à cidade de Santos, descrito na seção (7.3), foram utilizadas diferentes funções para este custo.

6.3.3.1 Custo por omissão constante para todas as viagens

A função mais simples adotada é aquela que considera o custo de omissão constante para todas as viagens. Desta forma são suprimidas viagens que levam à economia de veículos, independentemente das suas características de demanda de passageiros ou de suas inter-relações com as outras viagens do sistema. Neste caso o custo de omissão de uma viagem é dado por:

$$C_{oi} = K, \text{ para toda viagem } i \in N \tag{6.2}$$

onde K é uma constante adimencional controlada pelo usuário para regular o número de viagens suprimidas na programação. Ao assumir valores extremos para a constante K, são obtidas programações típicas, isto é, fazendo $K = \infty$ todas as viagens serão executadas. Por outro lado, assumindo $K = 0$ será gerada uma programação nula. Assim, o valor desta constante deve ser mantido em um patamar tal que o número de viagens suprimidas esteja de acordo com a espectativa do usuário.

6.3.3.2 Custo por omissão em função da taxa de ocupação do veículo

Para que fossem considerados fatores da demanda de passageiros às viagens, foi associado ao custo de omissão informações referentes à taxa de ocupação do veículo. O período de operação de cada linha foi dividido em intervalos caracterizados pela demanda, tais como: pré-pico, pico da manhã, pós-pico, entre-pico, etc. A taxa média de ocupação dos veículo em cada viagem foi incorporada ao seu custo de omissão, sendo utilizado também um multiplicador para calibrar os custos e regular o número de viagens omitidas. A expressão do custo de omissão para este caso é:

$$C_{oi} = D_i \times K, \text{ para toda viagem } i \in N \tag{6.3}$$

onde D_i representa a taxa média de ocupação do veículo durante a viagem i, e K é o seu fator multiplicativo, controlado pelo usuário.

6.3.3.3 Custo por omissão em função da taxa de ocupação do veículo e da qualidade da programação

Um fator que reflete a qualidade de um serviço de transporte público é o intervalo entre as partidas dos veículos nas respectivas linhas. Se por um lado o intervalo mais amplo entre duas partidas induz a uma alta taxa de ocupação do veículo, por outro lado ele eleva o tempo de espera do passageiro no ponto, comprometendo portanto a qualidade do serviço. Para que este tempo de espera fosse considerado ao eliminar uma viagem, ele foi incorporado ao custo de omissão das viagens. O modelo ficou então com o custo de omissão dado pelo produto entre a taxa de ocupação do veículo e o intervalo de tempo até a próxima partida, além do fator multiplicativo.

$$C_{oi} = T_i \times D_i \times K, \text{ para toda viagem } i \in N \tag{6.4}$$

onde T_i é o intervalo de tempo até a próxima partida na linha em relação à viagem i, D_i e K têm a mesma interpretação da expressão (6.3).

De uma forma geral, os custos de auto-atribuição devem ser controlados de acordo com o objetivo da aplicação. A constante adimensional K é utilizada neste sentido, ou seja, para calibrar os custos de auto-atribuição e, por conseguinte, o número de viagens excluídas da programação.

6.3.4 A Geração da Tabela de Horários Integrada à Programação dos Veículos

O modelo de circulação com arcos de auto-atribuição pode também ser utilizado para definir a tabela de horários das linhas de forma integrada com a programação dos veículos. Para que a tabela de horários mantenha uma dada qualidade e taxa de ocupação do veículo, para cada período de operação T (entre-pico, pré-pico, pico e pós-pico), define-se o número de viagens a serem realizadas N_T por:

$$N_T = \text{Máx}\left\{\frac{\text{Comprimento do Intervalo }_T}{\text{Tempo mínimo de espera }_{E_T}} \ , \ \frac{\text{Taxa de ocupação}}{\text{Capacidade do veículo}}\right\} \qquad (6.5)$$

O período da operação é discretizado, gerando um conjunto de viagens muito maior do que o número total de viagens que se deseja realizar. Considerando o conjunto de todas as possibilidades geradas, o modelo de otimização determina quais as viagens que devem ser executadas, tal que o custo total seja minimizado e a restrição de número de viagem N_T por período T seja satisfeita.

A idéia descrita acima pode ser modelada acrescentando-se ao modelo de circulação as restrições adicionais que assegurem a execução do número mínimo de viagens por linha e por período de operação. No entanto, o problema de fluxo em redes com restrições adicionais não pode ser atacado diretamente com o algoritmo *Out-of-Kilter*. Esta dificuldade pode ser contornada aplicando-se a técnica da Relaxação Lagrangeana, na qual as restrições adicionais são retiradas do conjunto de restrições do problema e as quantidades por elas violadas são incluídas como penalidades na função objetivo.

Considerando-se a existência de P períodos diferentes, define-se pela equação (6.5) o número mínimo de viagens N_T^p a serem executadas em cada período T_p, para $p = 1, ..., P$. O modelo de circulação com restrições adicionais fica:

$$\text{Min} \sum_{(i,j)\in A} c_{ij}x_{ij} \qquad \text{sujeito a} \qquad (6.6)$$

$$\sum_{j \in N} x_{ij} - \sum_{j \in N} x_{ji} = 0 \qquad \forall j \in N \tag{6.7}$$

$$\sum_{i \in T_p} (x_{i'i''} - x_{i''i'}) \geq N_T^p \qquad p = 1, ..., P \tag{6.8}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i, j) \in A - (s, r) \tag{6.9}$$

Relaxando as restrições (6.8) e incluindo-as como penalidade na função objetivo, conforme descrito no anexo (B), tem-se o Problema Lagrangeano:

$$\text{Min} \sum_{(i,j) \in A} c_{ij} x_{ij} + u[\sum_{p=1}^{P} (N_T^p - \sum_{i \in T_p} (x_{i'i''} - x_{i'i''}))] \qquad \text{sujeito a} \tag{6.10}$$

$$\sum_{j \in N} x_{ij} - \sum_{j \in N} x_{ji} = 0 \qquad \forall j \in N \tag{6.11}$$

$$x_{ij} \in \{0, 1\}, \ \forall \ (i, j) \in A - (s, r) \tag{6.12}$$

onde u é o vetor de multiplicadores de Lagrange e x é uma solução viável para o problema. O vetor u foi calculado pelo Método do Subgradiente, assumindo um limitante superior UB do valor ótimo da função objetivo Z^* e fazendo $\lambda = 0,7$, valor encontrado empiricamente.

Embora este modelo tenha sido implementado e testado com dados hipotéticos, não foi possível levantar as informações necessárias para realizar um estudo mais apurado acerca da qualidade das soluções geradas.

6.3.5 Planejamento Integrado Veículo – Tripulação

Trabalhando com o tempo de terminal requerido antes do início de cada viagem é possível gerar programações que também atendam às necessidades da tripulação. Para que esta estratégia fosse utilizada, implementou-se uma estrutura de dados que permite definir o tempo de terminal por faixa de horário em que a viagem é executada, para as diferentes linhas em estudo. Assim, tornou-se possível gerar programações controlando-se o parâmetro tempo de terminal que precede cada viagem. Na aplicação referente à cidade de Belo Horizonte, foram geradas programações com tempos de terminal mais amplos nas faixas de horário que abrangem a troca da tripulação. Este controle do tempo de terminal também foi utilizado para os horários em que o tempo previsto de viagem é muito apertado, gerando folgas que permitem um dado atraso do veículo, sem comprometer o cumprimento da tabela de horários.

6.3.6 Tratando a Variabilidade dos Tempos de Viagem

Uma vez que o tempo de viagem pode variar muito durante o período de operação, foi implementada uma estrutura de dados que permite definir uma variação no tempo de viagem tão acurada quanto desejado, ou seja, o tempo de viagem pode ser definido para intervalos de tempo tão pequenos quanto necessário. No estudo de caso referente à cidade de Belo Horizonte, foram informados os tempos de viagem nas respectivas linhas, para cada intervalo de 15 minutos. Desta forma foram geradas programações com tempos de viagem bastante realistas, facilitando sua execução.

6.4 Detalhes de Implementação

A versão do algoritmo *out-of-kilter* utilizada nos experimentos iniciais não dispunha de uma estrutura de dados adequada para o manuseio de problemas de fluxo em redes. Uma estrutura de dados adequada reduz consideravelmente a complexidade algoritmo, reduzindo consequentemente o tempo de processamento. Para representar uma rede é necessário armazenar dois tipos de dados:

a) a topologia da rede, isto é, os nós e estruturas dos arcos e

b) os dados técnicos tais como custo, capacidade dos arcos e a demanda nos nós.

A seguir são apresentadas as estruturas de dados mais difundidas para representar problemas de fluxo em redes direcionadas. Para maiores detalhes, ver Ahuja et al. (1993).

6.4.1 Matriz de Incidência Nó-Arco

Esta representação mantém uma matriz de dimensão $n \times m$, onde n é o total de nós e m o total de arcos, ou seja, com um linha para cada nó e uma coluna para cada arco. Nesta matriz, cada coluna possui apenas dois elementos não nulos, assumindo valor $(+1)$ na linha do nó origem e (-1) na linha correspondente ao nó destino do arco. Portanto, a matriz contém apenas $2m$ elementos não nulos de um total de nm elementos. Outra característica interessante desta matriz é que o número de

elementos não nulos em uma linha corresponde ao grau do nó, ou seja, ao número de arcos que estão ligados ao nó. Deste total, a soma de elemento iguais a $(+1)$ resulta no grau externo do nó, isto é, o número de arcos que partem do nó. Analogamente, o grau interno de cada nó que está associado aos elementos (-1) da linha, e que corresponde ao total de arcos que chegam no nó. Logo, o cálculo do grau interno ou externo de um nó requer m comparações, segundo esta representação.

Como a matriz de incidência armazena todos os seus elementos, esta não é uma representação eficiente do ponto de vista computacional. Sua importância se deve às propriedades matemáticas da estrutura da matriz. Uma segunda representação matricial relevante é apresentada a seguir.

6.4.2 Matriz de Adjacência Nó-Nó

A matriz de adjacência têm dimensão $n \times n$, utilizando uma linha e uma coluna para representar cada nó. Um arco (i, j) é representado pelo elemento a_{ij}, o qual assume valor 1 se o arco existe e 0 caso contrário. Para armazenar informações tais como custo e capacidade dos arcos, são necessárias duas outras matrizes de mesma dimensão.

Nesta representação apenas m elementos, dentre um total de n^2, são diferentes de zero, sendo portanto eficiente apenas quando se tratar de matrizes densas. Uma vantagem da rede de adjacência é a simplicidade de manuseio, pois o grau externo de um nó é igual ao número de elementos não nulos na linha correspondente ao nó. Logo, são necessárias n comparações para se obter o grau interno ou externo de um nó.

A seguir são apresentadas estruturas especiais que permitem encontrar o grau interno ou externo de um nó em tempo proporcional à magnitude destes conjuntos.

6.4.3 Listas de Adjacência

A *lista dos arcos adjacentes* a um nó é o conjunto de todos os arcos ligados ao nó. Da mesma maneira define-se a *lista dos nós adjacentes*. Esta representação utiliza uma

estrutura de lista de ligação simples para armazenar a lista dos nós de adjacência de cada nó. Uma lista de ligação simples é uma coleção de registros, contendo um ou mais campos cada um. No caso da lista dos nós adjacentes, ela é composta por registros, os quais representam e contém as informações dos arcos adjacentes. Cada um destes arcos aponta para o próximo arco adjacente ao nó em questão. Caso o arco seja o último da lista de adjacência, ele apontará para nulo. Esta representação necessita ainda de um conjunto de n apontadores, um para cada nó, indicando qual é o primeiro arco da lista de adjacência. Caso o nó não tenha arco adjacente, seu apontador indicará nulo.

6.4.4 Representação do tipo *Forward-Reverse*

Esta representação, assim como as listas de adjacência, guarda a lista dos nós adjacentes de cada nó da rede. Mas diferentemente da estrutura anterior, a técnica *forward-reverse* utiliza vetores para manter estas listas. Inicialmente os arcos são ordenados e numerados em uma seqüência tal que os primeiros arcos são aqueles que saem do nó 1, seguidos dos arcos que saem do nó 2 e assim por diante. Este é o vetor dos arcos ordenados pelo nó de origem, ou vetor dos *arcos de saída* dos nós (*forward list*). As informações dos arcos são carregadas na mesma ordem em quatro vetores que armazenam o nó origem, o nó destino, o custo e a capacidade do arco. Além da lista de arcos, existe um apontador $point(i)$ para cada nó i, o qual indica o arco de menor índice ligado ao nó i. Assim, os arcos que saem de um determinado nó i localizam-se no vetor de saída, nas posições que vão de $point(i)$ a $point(i+1)-1$. Caso $point(i)$ seja maior do que $point(i+1)-1$, o nó não tem arco saindo dele. De maneira análoga ordena-se os arcos pelo nó destino, criando-se o vetor dos arcos de chegado nos nós (*reverse list*) e os respectivos apontadores *rpoint(i)* que indicam o arco com menor índice que chega no nó i. Para percorrer todos os arcos que chegam em um determinado nó i basta percorrer o vetor dos arcos de chegada no intervalo que vai de $rpoint(i)$ a $rpoint(i+1)-1$.

6.4.5 A Estrutura de Dados Escolhida

Uma implementação eficiente das estruturas depende da linguagem de programação adotada. A estrutura de listas de adjacência é recomendável quando estiver sendo utilizada uma linguagem de programação que tem facilidade de manipulação de apontadores. Caso contrário deve ser utilizada a estrutura *forward-reverse* para representar o problema de fluxo em redes.

A representação *forward-reverse* requer menos espaço de armazenamento do que a lista de adjacência. Por outro lado, a estrutura de listas de adjacência requer menos esforço computacional para adicionar e retirar arcos da rede.

Uma vez que a técnica de redução dos arcos da rede que representa o problema de programação de veículos resolve uma seqüência de problemas de fluxo em rede, adicionando novos arcos a cada novo problema, foi escolhida a estrutura com maior capacidade de realizar tais operações, ou seja, a representação por listas de adjacência.

6.4.6 Resultados Alcançados

Os resultados mostram claramente a importância de se utilizar uma estrutura de dados adequada, visto que a redução no tempo de processamento foi em média de 75% do tempo requerido anteriormente à adoção da estrutura. O gráfico (6.3) mostra o tempo de processamento na resolução de um conjunto de problemas da cidade de Reading, com e sem a utilização de listas de adjacência .

6.5 Conclusões

A representação do PPV como um problema de circulação mostrou-se bastante eficiente no sentido de permitir as diversas flexibilizações descritas acima. Quando combinada com a técnica de geração de arcos e com o algoritmo *out-of-kilter*, a metodologia resultante permitiu a resolução de problemas de grande magnitude, além de suportar a inclusão de restrições adicionais e suas respectivas relaxações

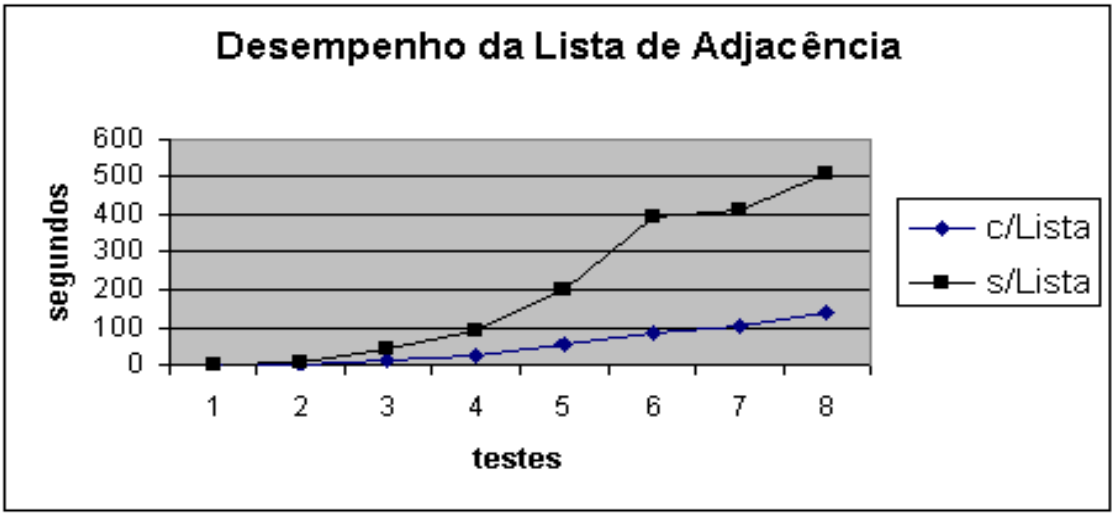

Figura 6.3: Comparação dos tempos de processamento.

lagrangeanas. Desta forma, foi possível resolver problemas de diferentes naturezas e peculiaridades.

Os resultados apresentados nos capítulos finais mostram a eficiência e aplicabilidade da metodologia proposta, sendo importante destacar que todos os problemas abordados foram resolvidos na sua otimalidade, inclusive alcançando resultados ligeiramente melhores do que aqueles alcançados por um sistema heurístico *BOOST* de renome internacional no meio acadêmico, pela sua tradição na abordagem do PPV.

Capítulo 7

Aplicações a Problemas Reais

7.1 Introdução

Os testes descritos no capítulo anterior apontaram o método de geração de arcos como sendo o mais eficiente tendo em vista o tempo de processamento na resolução do problema. Este método, que combina o modelo de pseudo-designação com a técnica de geração de arco, foi escolhido para dar continuidade ao trabalho, não somente pela sua eficiência computacional, mas também por permitir a inclusão de restrições adicionais. Assim, o método foi aprimorado considerando-se algumas restrições que surgem com freqüência na prática, tornando-o mais flexível e abrangente na abordagem de casos reais brasileiros.

Foram resolvidos problemas provenientes das cidades de *Reading* e de Sorocaba, com o objetivo de verificar a eficiência computacional do método e a validade das soluções obtidas, tendo como referência o método heurístico *BOOST*, largamente aplicado e conhecido no meio científico. Nestes casos, os dados processados, assim como o sistema computacional heurístico, foram fornecidos pelo grupo de programação de veículos e tripulação da Universidade de *Leeds*, com a qual o orientador deste trabalho mantém vínculos acadêmicos.

A maioria dos trabalhos encontrados na literatura visa desenvolver ou aprimorar modelos teóricos para solucionar o problema de programação de veículos, sem considerar a realidade na qual o mesmo está inserido. A tripulação por exemplo, gera restrições que devem ser obrigatoriamente satisfeitas, interferindo diretamente na

alocação ótima dos veículos. As restrições trabalhistas mais frequentes são:

a) limites mínimo e máximo para a jornada de trabalho,

b) tempo de refeição para jornadas a partir de determinada duração,

c) intervalo de descanso após um determinado número de horas,

d) tempo de troca de tripulação, etc.

Estas restrições têm sido tratadas no problema de programação da tripulação, o qual se torna de grande complexidade e tem sido abordado por diferentes técnicas (Daduna e Voß 2000, Wilson 1999, Daduna et al. 1995). Entretanto algumas destas restrições podem ser facilmente incorporadas ao problema de programação dos veículos, sem torná-lo mais complexo.

Nas aplicações referentes às cidades de Santos e de Belo Horizonte utilizou-se a estratégia de incorporar algumas restrições da tripulação à programação dos veículos. Assim, foi criado o parâmetro "tempo de terminal"após cada viagem para gerar programações contendo tempos livres nos terminais em determinadas faixas horárias e de diferentes durações, para satisfazer parte das restrições da tripulação. Neste sentido, foi estabelecido um contato direto com as empresas de transporte, facilitando o levantamento dos condicionantes operacionais e a avaliação das soluções encontradas. Os estudos dos problemas de Santos e Belo Horizonte tiveram como principal objetivo flexibilizar o modelo para gerar programações de interesse prático.

7.2 *Reading* e Sorocaba

Para verificar o valor prático das programações encontradas pelo método de geração de arco, um conjunto de problemas baseados em dados reais foram resolvidos, comparando então os resultados obtidos com aqueles fornecidos pelo sistema heurístico *BOOST*. Nesta comparação foram relacionados os custos totais, assim como as características operacionais das programações .

Os resultados obtidos estão resumidos nas tabelas (7.1) a (7.3), nas quais o método de geração de arcos é referenciado pelas inicias GA, e os tempo estão no formato

hh:mm. A coluna de "Viagens Mortas" mostra o tempo total de viagens mortas realizadas pelos veículos, e a coluna "Tempo de Espera" mostra o tempo total de espera dos veículos nos terminais. A coluna "Núm. de Retornos à Garagem (duração)" mostra o total de retornos temporários à garagem, com o tempo total de estacionamento temporário entre parênteses. O Custo Operacional Total, com os tempos em minutos, é computado pela expressão:

$$Custo\ Total = 2 \times Viagens\ Mortas + Tempo\ de\ Espera + (tempo\ mínimo\ de\ garagem) \times número\ de\ returnos \tag{7.1}$$

Esta expressão é utilizada no sistema *BOOST* e outros sistemas de programação de veículos, produzindo resultados de boa qualidade em vários casos reais. Entretanto, como *BOOST* é um sistema heurístico, esta expressão básica é suplementada por outras regras que tratam de situações nas quais uma função objetivo muito rígida não consegue avaliar os vários aspectos reais do problema. Em alguns casos a solução ótima não tem qualidades tão apreciáveis quanto soluções quase ótimas.

Uma dificuldade em alcançar uma função objetivo apropriada para o problema está em encontrar o equilíbrio para o número de retornos à garagem. Os retornos à garagem são importantes na solução, mas não podem ser muito freqüentes, caso contrário a programação terá muitas viagens mortas durante o período e, conseqüentemente, a duração de cada parada temporária será curta. Por esta razão foi adotado neste trabalho o período de 30 minutos para o tempo mínimo de garagem, que é um valor largamente utilizado no Reino Unido (comunicação pessoal Wren e Kwan 1999).

Outra dificuldade na definição de uma função objetivo compatível com a realidade é o impacto da programação dos veículos sobre a programação da tripulação. Para a programação dos veículos o tempo de espera nos terminais é mais barato do que o tempo de viagem morta; no entanto, para a tripulação ambos têm o mesmo custo, a menos que o veículo possa ser abandonado pela tripulação quando este estiver estacionado em algum terminal.

As dificuldades mencionadas acima podem ser melhor trabalhadas nos métodos heurís- ticos devido às suas flexibilidades. O sistema *BOOST*, por exemplo, não considera apenas o custo total na avaliação de um resultado, mas também leva em

conta a distribuição do tempo de trabalho entre viagens mortas, o tempo de espera nos terminais, e o número de retornos temporários à garagem. O mesmo não acontece com os métodos exatos, os quais têm como objetivo diminuir o custo total, sem levar em conta as outras características da solução.

Na prática a qualidade de uma programação vai depender da combinação dos seguintes atributos:

a) tempo total de viagens mortas - TVM,

b) tempo total de espera nos terminais - TT,

c) número de retornos à garagem - NRG, e

d) tempo total de estacionamento temporário na garagem - TG.

A coluna de custo total é portanto uma referência para verificar a qualidade da solução em relação à solução ótima, cuja função objetivo é definida pela expressão (7.1).

7.2.1 *Reading* – Reino Unido

Este testes foram realizados utilizando dados reais da cidade de *Reading* no Reino Unido. A seqüência de problemas foi criada aumentando-se o número de rotas incluídas na programação, de tal maneira que o número de veículos aumentasse progressivamente. Os problemas mais significativos são apresentados na tabela (7.1). Ambos os métodos apresentam os mesmos resultados para os três primeiros testes, mostrando que nestes casos as soluções coincidem. Porém no quarto teste o algoritmo de geração de arcos fornece uma solução com custo total ligeiramente melhor do que aquela fornecida pelo *BOOST*. Analisando as caracterísitcas desta solução, verifica-se que o algoritmo de geração de arcos tem 57 minutos a menos de viagem morta e 5 horas e 38 minutos a mais de tempo de espera nos terminais do que o sistema *BOOST*. Por outro lado a solução do *BOOST* tem 8 retornos à garagem a mais do que o algoritmo de geração de arcos, com um total de 4 horas e 40 minutos a mais de tempo de estacionamento temporário.

Tabela 7.1: Testes com dados da cidade de Reading - Reino Unido.

	Viagem Morta		Tempo de Terminal		Núm. de Retornos		Custo Total	
Carros	G.A.	Bost	G.A.	Bost	G.A.	Bost	G.A.	Bost
47	17:53	17:53	57:16	57:16	—	—	5.582	5.582
59	21:07	21:07	65:32	65:32	3 (03:45)	3 (03:45)	6.556	6.556
68	23:49	23:49	78:20	78:20	3 (03:45)	3 (03:45)	7.648	7.648
76	24:28	25:25	71:26	65:48	3 (03:45)	11 (08:25)	7.312	7.328
84	26:33	26:25	69:29	69:45	3 (03:45)	3 (03:45)	7.445	7.445

Tendo em vista a realidade operacional, a solução do sistema *BOOST* é considerada melhor do que a solução do método de geração de arcos, pois a tripulação recebe o mesmo valor por hora de trabalho, independente de se tratar de hora de viagem morta ou de espera nos terminais. Sendo assim, a solução heurística tem 4 horas e 51 minutos a menos de pagamento de tripulação por dia do que a solução exata fornecida pelo algoritmo de geração de arcos. Esta é uma situação em que a solução heurística quase ótima apresenta uma qualidade melhor do que a solução ótima pois leva em conta a programação da tripulação ao programar os veículos.

7.2.2 *Reading* com Preferência de Ligações

O caso estudado tem um total de 309 viagens distribuídas em 36 rotas, com preferência de ligação entre certos conjuntos de rotas. Inicialmente o problema foi resolvido sem levar em conta as preferências de ligação entre grupos de linhas. Posteriormente as preferências foram ativadas e o problema foi resolvido novamente. Foram consideradas onze diferentes preferências, cada uma delas definida através dos conjuntos de chegadas e de partidas que variam de uma a dez rotas. O valor adotado para o redutor do custo de ligação foi de quinze minutos e a tabela abaixo sintetiza os resultados obtidos.

No primeiro caso, sem as preferências de ligação, o método de Geração de Arcos produziu uma programação mais barata, embora tenha satisfeito espontaneamente um número menor de ligações preferenciais. Uma vez ativada as preferências, o algoritmo de geração de arcos efetuou um número maior de ligações preferenciais,

Tabela 7.2: Soluções sem as preferências de ligação e com preferências de ligação respectivamente.

	Viagem Morta		Tempo de Terminal		Núm. de Retornos		Custo Total	
Carros	G.A.	Bost	G.A.	Bost	G.A.	Bost	G.A.	Bost
15	05:09	05:11	17:01	16:59	96	97	1.639	1.641
15	05:09	05:09	17:08	17:01	105	104	1.646	1.639

embora o custo total tenha sido maior do que no caso do sistema *BOOST*.

Foi verificado um desempenho qualitativo superior do método exato em relação ao *BOOST*, pois o primeiro produziu um resultado mais barato quando não existiam preferências de ligação. E um resultado com maior número de ligações preferencias satisfeitas quando estas foram requeridas. O sistema *BOOST* produziu resultados teoricamente incoerentes, pois o problema mais restrito tem custo total menor do que o problema mais relaxado.

7.2.3 Sorocaba

O conjunto de dados da cidade de Sorocaba é consideravelmente maior do que os da cidade de *Reading*, tendo sido estudado previamente por Wren e Gualda (1999). O maior teste realizado com os dados de *Reading* (tabela 7.1) tem 1.518 viagens, exigindo no mínimo 84 veículos, enquanto o maior teste da cidade de Sorocaba trata de 2.732 viagens, que necessitam de no mínimo 111 veículos. A rede montada para representar o maior problema da cidade de Sorocaba é composta por 5.466 nós e 3.379.000 arcos. Para resolver problemas desta magnitude é imprescindível a aplicação de técnicas de redução dos arcos da rede, sem as quais o problema se torna intratável do ponto de vista computacional.

No primeiro problema, as soluções produzidas pelos dois métodos coincidem em custo total e qualidade. No entanto, para o segundo problema o algoritmo de geração de arcos encontra uma solução com menor custo total, com 51 minutos a menos de viagem morta e 52 minutos a mais de tempo de espera nos terminais. As soluções apresentam praticamente mesmo custo com tripulação (viagem morta mais espera nos terminais) e igual número de retornos à garagem, porém a solução do sistema

Tabela 7.3: Testes com dados da cidade de Sorocaba - Brasil.

	Viagem Morta		Tempo de Terminal		Núm. de Retornos		Custo Total	
Carros	G.A.	Bost	G.A.	Bost	G.A.	Bost	G.A.	Bost
107	122:08	122:08	124:48	124:48	23 (165:13)	23 (165:09)	22.834	22.834
111	94:42	95:33	101:19	100:27	28 (191:52)	28 (192:18)	18.283	18.333

BOOST conta com 36 minutos a mais de estacionamento temporário.

7.2.4 Tempo de Processamento

O sistema *BOOST* é um módulo do pacote *OpenBus* que realiza a programação de veículos e tripulação. Sua execução não se dá isoladamente, portanto o tempo de processamento da solução do PPV envolve também o tempo de processamento da interface. Embora este processamento tenha sido minimizado durante a realização dos testes, os dados apresentados nas tabelas (7.4) e (7.5) referentes ao *BOOST* correspondem a aproximações dos tempos de execução do método heurístico. Os tempos são fornecidos nas unidades minutos:segundos (mm:ss).

Tabela 7.4: Tempos de processamento para alguns casos de *Reading*.

Método	caso 1	caso 2	caso 3	caso 4	caso 5	caso 6	caso 7	caso 8
G. A.	00:07	00:24	02:33	05:35	12:48	22:10	26:13	30:36
BOOST	00:13	00:45	02:42	07:41	12:19	09:41	12:39	20:35

Tabela 7.5: Tempos de processamento para os casos de Sorocaba.

Método	caso 1	caso 2
G. A.	22:10	22:34
BOOST	2:47	6:50

Como pode ser observado, o algoritmo de geração de arcos exigiu entre 0,5 e 1,5 do tempo de processamento do *BOOST* nos testes com os dados de *Reading* e entre 3,2 e 8 vezes o tempo requerido pelo *BOOST* para os testes com a cidade de Soroca-

ba. Entretanto, para estas magnitudes de tempo, tais diferenças não levam a um comprometimento na utilização do método exato.

7.2.5 Conclusões da Aplicação *Reading* − Sorocaba

Na maioria dos casos estudados, o método heurístico e o algoritmo de fluxo em redes encontraram a mesma solução, sendo que, para alguns problemas de maior porte, o algoritmo de fluxo em redes encontrou o ótimo global que o método heurístico não encontrou. Considerando o custo total definido pela expressão (7.1), o método heurístico apresentou soluções com um erro de 0,22% em relação ao ótimo no caso da cidade de *Reading*, e de 0,27% no caso da cidade de Sorocaba. Embora as diferenças não sejam grandes, deve ser lembrado que as programação são diárias. Logo, uma pequena diferença pode acarretar em um economia significativa ao final do período de um ano.

Este estudo demostrou que a metodologia proposta, embora não apresente o mesmo desempenho computacional do sistema heurístico, é capaz de resolver problemas de grandes magnitudes, inclusive gerando soluções melhores do que aquelas encontrar pelo *BOOST*. Além disso, o modelo ainda comporta restrições adicionais, o que o capacita a resolver problemas com características particulares, como é o caso de preferência de ligações.

Logo, pode-se concluir que a combinação da técnica de geração de arcos com o modelo de pseudo-designação e o algoritmo *Out-of-Kilter* representa uma ferramenta eficiente na resolução de problemas práticos e de grande porte que envolvem a programação de veículos no sistema de transporte público.

7.3 Santos

Quanto aos estudos de caso, foram feitos levantamentos dos dados e procedimentos operacionais adotados pelas empresas envolvidas, assim como a forma de gestão do sistema de transporte público no referido município. Estes contatos com as empresas foram primordiais para o entendimento da operação e conseqüente aprimoramento

do modelo. No caso da empresa que opera na cidade de Santos, foi levantada a situação descrita abaixo.

7.3.1 Situação Encontrada

O sistema de transporte "Seletivo"de Santos é um serviço diferenciado por apresentar as seguintes características:

a) os veículos são micro-ônibus (28 assentos) equipados com ar condicionado,

b) os passageiros só viajam sentados,

c) o motorista também desempenha a função de cobrador,

d) os veículos cumprem suas tabelas de horários com rigidez, e

e) o preço da passagem é mais alto do que no transporte coletivo comum.

O serviço é composto de oito linhas circulares, partindo de três terminais distribuídos em diferentes pontos da cidade de Santos. A tabela (7.6) sintetiza as informações deste serviço para os dias úteis (segunda a sexta-feira).

Tabela 7.6: Dados gerais da operação na ocasião do estudo.

Linha	Terminal	Num. Veículos	Num. Viagens
201	Ponta da Praia	9	72
202	Ponta da Praia	7	63
204	Centro	3	32
205	Zona Noroeste	5	42
206	Ponta da Praia	2	24
207	Ponta da Praia	2	24
208	Ponta da Praia	3	36
222	Zona Noroeste	3	24
Total	3	34	452

A operação em curso apresenta as seguintes características:

- Para cada linha é definido o número de veículos que nela atua, tendo em vista a demanda na linha e o tamanho da frota disponível.

- Cada veículo opera somente em uma determinada linha.
- O quadro de horários das linhas é definido dividindo-se o tempo de duração da viagem pelo número de veículos que atuarão na linha. São feitos os arredondamentos necessários para facilitar a operação.
- Os tempos de viagem são tidos como constantes.
- Ao final de cada viagem é dado um "tempo de ponto" de 10 minutos para o descanso do motorista.

Tabela 7.7: Dados da programação atual.

Total de viagens mortas - (TVM)	31:50
Tempo de espera no terminal - (TET)	69:40
Tempo na garagem - (TG)	00:00
Retornos à garagem - (NRG)	0
Número de veículos - (NV)	34
Custo operacional	8.000

Os recursos dispendidos na programação corrente, descritos na tabela (7.7), foram utilizados como parâmetro para avaliar as programações alternativas encontradas. Analisando suas características operacionais, conclui-se que:

- Não é considerada a variação da demanda nas linhas, visto que a freqüência de partida dos veículos é constante no decorrer do dia.
- Ocorre uma super-utilização dos veículos no período de pico e uma sub-utilização no período de entre-pico.
- Pela forma de definição do quadro de horários, existe uma pequena margem de otimização.
- Não existe retorno de veículos à garagem no período de entre-pico.
- Este serviço segue um padrão operacional de linhas de baixa demanda.

Nesta aplicação foram exploradas as diversas possibilidades de flexibilização do modelo, gerando soluções que variassem consideravelmente tanto no custo total quanto

nas características operacionais. Assim, foram realizados dois experimentos diferentes tais que:

a) inicialmente todas as viagens planejadas fossem rigorosamente cumpridas, e

b) no segundo caso, que o modelo permitisse que algumas viagens fossem excluídas do quadro de horários.

Para permitir a omissão de viagens na programação resultante, foi utilizado o modelo de pseudo-designação com arcos de auto-atribuição descrito anteriormente. Nesta versão do modelo, a retirada de um conjunto de viagem do quadro de horários na programação resultante está associada à redução na frota em operação.

A seguir são apresentados os principais resultados alcançados, ou seja, apenas aqueles que mais se aproximaram da situação atual.

7.3.2 Realizando todas as Viagens Planejadas

As programações geradas nesta etapa do estudo atendem rigorosamente a todas as viagens planejadas, respeitando os seus respectivos horários e locais de partida e de chegada. Os dados básicos que alimentaram o sistema foram: o total de viagens planejadas, os horários e locais de início e de término das viagem, o tempo de viagem entre os terminais, e o tempo de viagem entre cada terminal e a garagem, estando o veículo fora de operação. Também é informado o tempo mínimo de permanência na garagem, caso o veículo retorne a ela entre duas jornadas de trabalho. Foi adotado para este último parâmetro o período mínimo de 30 minutos de estacionamento temporário na garagem.

7.3.2.1 Sem troca de linhas durante a operação

Nestes casos não foi permitido que os veículos atuassem em mais de uma linha durante o período de trabalho. Quanto ao tempo mínimo de permanência no terminal foram testadas duas situações: i) sem descanso, isto é, zero minutos de terminal, e ii) com dez minutos de descanso ao final de cada viagem. As programações encontradas são apresentadas na tabela (7.8). Para avaliar como estão distribuídos os

Tabela 7.8: Programações sem troca de linha e tempos mínimos de terminal de 0 e 10 minutos.

Tempo de Terminal	Programação 1 0 minutos		Programação 2 10 minutos	
	hh:mm	variação	hh:mm	variação
TVM	30:50	− 01:00	31:50	00:00
TET	44:50	− 24:50	69:40	00:00
TG	00:00	00:00	00:00	00:00
NRG	0	0	0	0
NV	32	− 2	34	0
Custo	6.390	− 20,13%	8.000	0%

tempos de terminal nas programações encontradas, foi contruído o histograma com diferentes folgas nos terminais e o números de viagens que respeitam estas folgas (tabela 7.9). A primeira classe é das viagens sem qualquer folga no terminal, a segunda classe é das viagens cuja folga é maior do que zero e menor ou igual a 5 minutos, e assim por diante.

Tabela 7.9: Histograma com número viagens e respectivas folgas nos terminais.

Folga (em min.)	0	5	10	15	20	25	40	60	120	180	240
Viagens na Prog1	200	0	205	1	5	0	0	9	0	0	0
Viagens na Prog2	0	0	420	0	0	0	0	0	0	0	0

A primeira programação da tabela (7.8) executa todas as viagens programadas utilizando dois veículos a menos na operação do que na situação atual, atingindo uma redução de 20,13% no custo operacional. Se por um lado ela não garante o tempo para o descanso do motorista ao final de cada viagem, por outro lado ela conta com 44 horas e 50 minutos livres nos terminais para criar condições de trabalho aos motoristas. Segundo o histograma de folgas, 200 viagens não apresentam tempo livre nos terminais, entretanto as 220 viagens restantes propiciam um descanso ao motorista de no mínimo 10 minutos. A segunda programação corresponde à situação em operação. Nesta solução cada viagem tem um descanso de dez minutos ao final do ciclo.

7.3.2.2 Com troca de linhas durante a operação

Esta é uma situação de maior flexibilidade onde um veículo pode realizar viagens de diferentes linhas em um mesmo dia de trabalho. Para que as soluções mantivessem características semelhantes às da programação atual, o sistema foi ajustado para evitar um alto índice de troca de linha durante a operação.

Assim como no caso anterior, foram testadas duas situações diferentes em função do tempo de terminal: *i*) sem garantia de descanso no terminal, isto é, mínimo de zero minutos de terminal, e *ii*) com um mínimo de dez minutos de terminal ao final de cada viagem. As programações encontradas são apresentadas na tabela (7.10) e o seu histograma de folgas é apresentado na tabela (7.11).

Tabela 7.10: Programações com troca de linha e tempos mínimos de terminal de 0 e 10 minutos.

Tempo de Terminal	Programação 1 0 minutos		Programação 2 10 minutos	
	hh:mm	variação	hh:mm	variação
TVM	32:40	+ 00:50	31:50	00:00
TET	31:20	− 38:20	69:40	00:00
TG	02:55	+ 02:55	00:00	00:00
NRG	1	+1	0	0
NV	32	− 2	34	0
Custo	5.830	− 27,13%	8.000	0%

Tabela 7.11: Histograma com número viagens e respectivas folgas nos terminais.

Folga	0	5	10	15	20	25	40	60	120	180	240
Viagens na Prog1	320	59	18	5	8	7	2	1	0	0	0
Viagens na Prog2	0	0	420	0	0	0	0	0	0	0	0

A primeira solução encontrada, embora opere com uma redução de dois veículos na frota e atinja uma redução de 27,13% em relação ao custo operacional vigente, não se mostra atraente por não permitir o descanso adequado do motorista em 76,19% das viagens, como pode ser observado na tabela (7.11). Na programação 2, embora seja assegurado o descanso do motorista, não existe redução no número de veículos em relação à programação atual, ocorrendo ainda a desvantagem da troca de linha

dos veículos durante a operação. Na programação 1 os veículos atuam em média em quatro linhas diferentes, e na programação 2 em média em três linhas distintas.

7.3.3 Omitindo Viagens Planejadas

Foi observado que as programações geradas nesta seção que reduzem a dimensão da frota levam a outros problemas operacionais de difícil solução, sendo portanto de pouco interesse prático. Surge então a necessidade de se gerar programações alternativas. Este objetivo foi atingido permitindo a supressão de algumas viagens para que novos quadros de interesse fossem produzidos. Para gerar outras alternativas de programação para o sistema de transporte, o modelo foi alterado de forma a considerar a possibilidade de não realizar algumas viagens, levando a uma redução da frota em operação.

A redução da frota pela omissão de algumas viagens está associada à relação custo/benefício no sistema como um todo. Neste sentido, foram atribuídos altos custo de omissão às viagens de maior interesse para o sistema. Às viagens consideradas de menor importância, atribuíram-se custos de omissão mais baixos.

Todas as programações geradas nesta etapa contemplam o descanso do motorista de 10 minutos após o cumprimento de cada viagem, como ocorre na operação atual. São apresentadas neste trabalho apenas as programações que mostram uma redução da frota, e que ao mesmo tempo não sejam muito diferentes da programação vigente, como forma de manter a qualidade no serviço.

A seguir são descritas as diferentes funções de custo atribuídas à omissão das viagens, juntamente com os resultados proporcionados pelo modelo de pseudo-designação com auto-atribuição.

7.3.3.1 Custo por omissão constante para todas as viagens

Inicialmente foi considerado um custo de omissão constante para todas as viagens. Desta forma foram suprimidas viagens que levam à economia de veículos, independentemente das suas características de demanda de passageiros ou de suas inter-

relações com as outras viagens do sistema. Logo, o custo de omissão de uma viagem é dado por:

$$C_{oi} = K, \text{ para toda viagem } i \in N \tag{7.2}$$

onde K é uma constante adimencional controlada pelo usuário para regular o número de viagens suprimidas na programação.

A programação apresentada na tabela (7.12) considera a operação que permite aos veículos realizar viagens em diferentes linhas em um mesmo dia de trabalho.

Tabela 7.12: Programação com custo constante por omissão de viagem.

	hh:mm	variação
TVM	29:50	− 02:00
TET	67:20	− 02:20
TG	00:00	00:00
NRG	0	0
NV	32	− 2
Custo	7.620	− 4,75%

Ao assumir um custo de omissão constante para todas as viagens, foram retiradas 16 viagens da programação, sendo 8 da linha 206 e 8 da linha 207. Assim, foi possível reduzir dois veículos na frota em operação, um de cada linha afetada. Algumas viagens das linhas 206 e 207 foram incorporadas a dois veículos da linha 202, que são os únicos a atuarem em linhas diferentes durante a operação. Estes dois veículos operam na linha 202 até por volta das 19:00, quando então realizam um bloco de quatro viagens das linhas 206 e 207, respectivamente.

As viagens suprimidas estavam distribuídas ao longo do dia, abrangendo períodos de pico e fora do pico. Resultado semelhante foi gerado para o custo de omissão dado em função da taxa de ocupação dos veículos (primeiro da tabela 7.13).

7.3.3.2 Custo por omissão em função da taxa de ocupação do veículo

Para serem considerados fatores da demanda de passageiros relativos às viagens, foi associado ao custo de omissão informações referentes à taxa de ocupação do veículo. O período de operação de cada linha foi dividido em intervalos caracterizados pela

demanda, tais como: pré pico, pico da manhã, pós pico, entre pico, etc. A empresa estudada forneceu a taxa média de ocupação dos veículo por viagem nos respectivos intervalos. Esta taxa, que reflete a demanda na linha, foi incorporada ao custo de omissão da viagem. Foi utilizado ainda um multiplicador para calibrar os custos e regular o número de viagens omitidas, como pode ser verificado na expressão abaixo.

$$C_{oi} = D_i \times K, \text{ para toda viagem } i \in N \tag{7.3}$$

onde D_i representa a taxa média de ocupação do veículo para a viagem i, e K tem a mesma interpretação da expressão (7.2).

Considerando o custos por omissão (7.3), não foi possível que o modelo gerasse programações alternativas sem que houvesse alguma troca de linha durante a operação. Portanto, as programações apresentadas realizam pelo menos uma troca de linha.

Tabela 7.13: Programações com custo por omissão em função da taxa média de ocupação dos veículos por período.

Consequências da redução de viagens		Veículos com troca de linha		
Tot. Veículos / Tot. Omissões	Linha/Omissões período das omissões	Linha principal	Linha sec.× num. viagens	Período viag. secundárias
32/16 Programação 1	206/8 viagens 07:30 - 17:25	202	206 × 4	fim período
	207/8 viagens 07:50 - 17:45	202	207 × 4	fim período
32/17 Programação 2	208/3 viagens 07:05 - 09:45	201	206 × 3	fim período
	207/4 viagens 07:50 - 12:05	202	206 × 4	fim período
	206/10 viagens 11:05 - 18:10	207	206 × 4	iníc.período
		208	206 × 3	iníc.período
33/8 Programação 3	206/8 viagens 07:30 - 17:25	202	206 × 4	fim período
33/8 Programação 4	207/4 viagens 07:50 - 12:05	202	206 × 4	fim período
	206/4 viagens 13:10 - 17:25	207	206 × 4	iníc.período

As duas primeiras programações na tabela (7.13) utilizam 32 veículos deixando de realizar 16 e 17 viagens respectivamente. Na primeira programação as oito viagens omitidas da linha 206 estão distribuídas no intervalo das 07:30 às 17:25. Neste caso,

apenas dois veículos operam em duas linhas diferentes. Estes dois veículos têm a linha 202 como principal, realizando um bloco de apenas 4 viagens nas linhas 206 e 207 respectivamente, no final do período.

Na segunda programação quatro veículos atuam em duas linhas distintas durante a operação, sempre em blocos de 3 a 4 viagens no início ou no final do período. Esta programação suprime 17 viagens e, portanto, uma a mais do que a programação anterior. A vantagem desta opção é que as viagens omitidas se concentram em períodos que não são considerados horário de pico das respectivas linhas.

As duas últimas programações da tabela (7.13) utilizam 33 veículos, omitindo em ambos os casos 8 viagens. Na terceira programação, todas as viagens suprimidas pertencem à linha 206, abrangem horários de pico e fora do pico, e um único veículo da frota atua em linhas diferentes. Este veículo realiza um bloco de 4 viagens na linha secundária no final do período. Na última programação apresentada, as viagens não realizadas pertencem às linhas 206 e 207, não se encontram em horários de pico, e dois veículos atuam em linhas diferentes, realizando blocos de 4 viagens na linha secundária no início e no final do período.

7.3.3.3 Custo por omissão em função da taxa de ocupação do veículo, do intervalo até a próxima viagem

Outro fator que reflete a qualidade de um serviço de transporte público é o intervalo entre as partidas dos veículos nas diversas linhas. Se por um lado o intervalo mais amplo entre duas partidas garante uma alta taxa de ocupação do veículo, por outro lado ele eleva o tempo de espera do passageiro no ponto. Para que este tempo de espera fosse considerado ao eliminar uma viagem, ele foi incorporado ao custo de omissão das viagens. E o modelo ficou com o custo de omissão dado pelo produto entre a taxa de ocupação do veículo e o intervalo de tempo até a próxima partida, além do fator multiplicativo dado por:

$$C_{oi} = T_i \times D_i \times K, \text{ para toda viagem } i \in N \tag{7.4}$$

onde T_i é o intervalo de tempo até a próxima partida na linha em relação à viagem i, D_i e K têm a mesma interpretação da expressão (7.3).

A tabela abaixo mostra as programações resultantes tendo em vista os custos de omissão expressos em (7.4).

Tabela 7.14: Programações considerando a taxa de ocupação do veículo e o intervalo até a próxima partida na omissão de uma viagem.

Consequências da redução de viagens		Veículos com troca de linha		
Tot. Veículos / Tot. Omissões	Linha/Omissões período das omissões	Linha principal	Linha sec.× num. viagens	Período viag. secundárias
32/19	208/9 viagens	208	206 × 4	iníc. período
Programação 1	07:05 - 12:25	208	204 × 5	iníc. período
	204/10 viagens	202	208 × 1	iníc. período
	13:05 - 18:10			
33/9	208/4 viagens	202	208 × 1	iníc. período
Programação 2	08:00 - 12:00			
	204/5 viagens	208	204 × 5	iníc. período
	13:05 - 17:45			

Para reduzir a frota ao mesmo número de veículos (32 e 33) das programações apresentadas no caso anterior, as programações desta seção suprimiram um número maior de viagens. No primeiro caso, para atingir uma frota com 32 veículos , foi necessário suprimir 19 viagens. Portanto 3 e 2 viagens a mais do que nas duas primeiras programações da tabela (7.13). No segundo caso, foi suprimida apenas uma viagem a mais. O número de veículos que fazem troca de linhas não difere muito em relação ao caso anterior.

7.3.4 Conclusões da Aplicação em Santos

As situações que garantem o descanso do motorista (Programação 2 nas tabelas 7.8 e 7.10) coincidem com a programação atual, sendo que atualmente os veículos são cativos às linhas. Isto se deve pelo fato do horário de partida das linha ter sido calculado dividindo-se o tempo de ciclo pelo número de veículos a operar na respectiva linha. Esta prática mostra que, neste município, a empresa de transporte é quem determina o quadro de horários, o qual é feito de acordo com suas conveniências. Isso pode ser justificado pelo carater seletivo do serviço prestado. Matematicamente, esta é uma situação na qual partiu-se da solução ótima (o número mínimo de veículos a ser utilizado por linha e tempo de terminal após cada viagem) para

montar o problema (o quadro de horário das viagens).

Nos casos em que o descanso do motorista não foi imposto ao modelo (Programação 1 nas tabelas 7.8 e 7.10) houve uma diminuição da frota em operação. Entretanto, soluções deste tipo não são facilmente aceitas devido às dificuldades encontradas pelas empresas para i) definir uma alocação adequada dos motoristas, ii) controlar as responsabilidades sobre os veículos durante a operação, iii) controlar a roleta, etc.

A possibilidade de não executar todas as viagens programadas permite gerar programações alternativas com diferentes graus de redução na frota. Os resultados mais interessantes do ponto de vista operacional são aqueles que refletem a taxa de ocupação dos veículos e os que combinam a taxa de ocupação com intervalo até a próxima viagem para definir o seu custo de omissão.

A tabela (7.13) apresenta dois grupos de programações com a mesma taxa de redução da frota, porém com diferentes características operacionais. Enquanto na primeira e terceira programações algumas das viagens omitidas pertencem ao horário de pico das respectivas linhas, na segunda e quarta programações as viagens suprimidas pertencem ao horário de entre-pico. Por outro lado, na primeira e terceira programações existe um número menor de veículos operando em linhas diferentes do que na segunda e quarta programações.

Embora os resultados apresentados na tabela (7.14) suprimam algumas viagens a mais do que nos casos anteriores para as mesmas dimensões da frota, adotando-se a função de custo de omissão de viagens (7.4) foi possível gerar outras alternativas de programação. A primeira e segunda programações da tabela (7.14) podem ser comparadas à segunda e quarta programações da tabela (7.14), respectivamente, pois elas têm a mesma frota e todas as viagens omitidas estão fora do horário de pico. A diferença qualitativa entre estes resultados, além do número de viagens omitidas, está nas linhas que sofrem redução de viagem, no número de veículos que atuam em linhas diferentes,linhas primárias e secundárias, além do número viagens realizadas nas linhas secundária.

Comparando as programações com 32 veículos nas duas últimas tabelas, observa-se que: i) 4 e 3 veículos atuam em linhas diferentes nas programações da respectivas

tabelas, e *ii*) são realizadas 14 e 12 viagens nas linhas secundárias nas duas programações, respectivamente. Para as programações com 33 veículos observa-se que: *i*) as programações omitem viagens pertencem a diferentes linhas, e *ii*) são realizadas 8 e 6 viagens nas linhas secundárias nas respectivas programações.

Assim, foi possível encontrar resultados alternativos e até complementares, para uma mesma dimensão da frota. As limitações nas margens de otimização impostas por este caso levaram à flexibilização do modelo, tornando-o capaz de gerar programações com diferentes frotas mínimas e opções de redução no quadro de viagens do sistema de transporte.

7.4 Belo Horizonte

Em contatos preliminares foram levantadas as características da operação da empresa e coletados os dados necessários à aplicação do modelo. Uma breve descrição dos dados correspondentes à situação atual é apresentada abaixo.

7.4.1 Colocação do Problema

Foram estudadas 9 linhas de uma Empresa que operam na cidade de Belo Horizonte. Estas linhas estão divididas em dois grupos com diferentes características. As informações abaixo se referem à operação de tais linhas em dias úteis.

7.4.1.1 Linhas Perimetrais

Corresponde ao atendimento bairro a bairro, sem necessariamente passar pelo centro da cidade. Abaixo são apresentadas as linhas com esta características, juntamente com seus pontos extremos, denominados pontos de controle, ou terminais. O ponto de controle 1 (PC1) é aquele onde a operação deve ser iniciada, porém este nem sempre corresponde ao ponto mais próximo à garagem. O ponto de controle 2 (PC2) corresponde ao outro extremo da linha, também dito ponto de virada. Para facilitar a identificação dos veículos pelos usuários, a BHTrans sugere que os ônibus que operam em um determinado tipo de linha tenham uma coloração específica. Os

veículos que operam nas linhas perimetrais são identificados pela cor amarela.

Tabela 7.15: Dados das linhas Perimetrais consideradas no estudo.

Número da Linha	Ponto de controle 1	Ponto de controle 2	Número de veículos	Número de viagens
1170	Santa Lúcia	(circular)	6	58
2152	Salgado Filho	Cruzeiro	7	98
4150	Shopping Del Rey	BH Shopping	10	128

7.4.1.2 Linhas Diametrais Interbairros

Estas linhas atendem às regiões periféricas, ligando-as ao centro. Assim como nas linhas perimetrais, o ponto de controle 1 (PC1) é aquele onde a operação deve ser iniciada, e o ponto de controle 2 (PC2) corresponde ao outro extremo da linha. Os veículos que operam nas linhas diametrais apresentadas na tabela (7.16) têm coloração azul.

Tabela 7.16: Dados das linhas Interbairros estudadas.

Número da Linha	Ponto de controle 1	Ponto de controle 2	Número de veículos	Número de viagens
8001A	Ana Lúcia	BH Shopping	12	160
8001B	Ana Lúcia	Fac. Milton Campos	13	156
8203	Renascença	Buritis	9	130
8207	Maria Goretti	Estrela Dalva	14	158
8208	Uni-Estoril	(circular)	9	79
9206	Vera Cruz	Buritis	14	158

Os pontos extremos PC1 e PC2 associados a cada uma das linhas são denominados genericamente de terminais, independente da estrutura encontrada no local.

A operação do serviço em curso apresenta as seguintes características.

- Para cada linha é definido o número de veículos que nela atua, tendo em vista a freqüência de partida na linha.
- Cada veículo opera em uma determinada linha.

- A definição do quadro de horários é feita pela empresa gestora do transporte público no município, a BHTRANS que utiliza como critério a demanda de passageiros e a taxa de ocupação dos veículos.
- Os tempos de viagem fornecidos pela empresa apresentam variações ao longo do dia, refletindo horários de pico e entre-pico que ocorrem na prática.
- A empresa assegura um tempo livre de cinco minutos sempre que o veículo atinge o PC2.

A partir dos dados da operação em curso foram extraídos os parâmetros necessários para o cálculo do custo variável da programação atual. Embora o tempo que o veículo permanece estacionado temporariamente na garagem não seja utilizado para compor o custo da programação, ele auxilia na análise da qualidade da solução. Estes dados se encontram na tabela abaixo.

Tabela 7.17: Parâmetros para avaliação do custo da programação atual.

Total de viagens mortas - TVM	43:40
Tempo de espera no terminal - TET	156:55
Tempo na garagem - TG	184:07
Retornos à garagem - NRG	28
Número de veículos - NV	94
Custo variável total	15.495

Analisando a situação atual, conclui-se que:

- O fato de a BHTRANS ser a responsável pelo quadro de horários, e não a empresa prestadora do serviço, torna a situação mais propícia à otimização. Por outro lado, um estudo prévio das linhas realizado pela BHTRANS fornece à empresa uma opção de programação com frota mínima.
- Existe retorno temporário de veículos à garagem no período de entre-pico.
- Este serviço segue um padrão operacional de linhas de alta demanda.

Nesta aplicação foi imposto ao modelo que todas as viagens planejadas deveriam ser executadas. Foram geradas diferentes alternativas variando os seguintes parâmetros operacionais:

a) Sem permissão de troca de linha, permissão de troca de veículos entre linhas do mesmo tipo: perimetrais ou diametrais, e permissão de troca entre todas as linhas.

b) Variação do tempo mínimo de terminal em 0, 3 e 5 minutos, e local de sua ocorrência, se no PC1 ou no PC2.

Os testes realizados, contemplando as diferentes formas de operação, não conseguiram diminuir a frota em operação; no entanto, foram encontradas programações alternativas com custos operacionais inferiores à programação vigente. Estes resultados estão agrupados abaixo, de acordo com as diferentes permissões de troca de linha.

7.4.2 Sem Troca de Linha

Nesta primeira situação foi mantida a mesma prática operacional vigente. Neste sentido impôs-se ao modelo que cada veículo permanecesse cativo a uma única linha durante todo o dia. Embora esta situação seja a menos favorável à otimização do sistema, o modelo produziu resultados de grande interesse para a empresa, pois as programações geradas estão de acordo com a prática operacional e ainda apresentam uma considerável redução de custo em relação à situação atual. Abaixo são apresentados os principais resultados com diferentes tempos de terminal e locais de ocorrência destes tempos. Também é apresentada a "variação"da solução em relação à programação corrente.

7.4.2.1 Tempo mínimo de terminal no PC1

Exigindo-se a ocorrência de 0, 3 e 5 minutos como tempos mínimos livres no terminal PC1 para todas as linhas e independente do período do dia (pico, fora-pico), foram obtidos os seguintes resultados:

Embora as programações com tempos mínimos nos terminais iguais a zero minutos não sejam de interesse prático, esta situação foi mantida como um referencial da situação ótima global.

Tabela 7.18: Programações sem troca de linha e tempos mínimos no PC1 de 0, 3 e 5 minutos.

Tempo no PC1	Programação 1 0 minutos		Programação 2 3 minutos		Programação 3 5 minutos	
	hh:mm	variação	hh:mm	variação	hh:mm	variação
TVM	46:25	+ 02:45	48:05	+ 04:25	48:05	+ 04:25
TET	99:00	− 57:55	105:52	− 51:03	108:15	− 48:40
TG	223:12	+ 39:05	227:33	+ 26:43	225:44	+ 41:37
NRG	33	+ 5	36	+ 8	36	+ 8
NV	94	0	94	0	94	0
CUSTO	12.500	− 19,3%	13.202	− 14,8%	13.345	− 13,9%

7.4.2.2 Tempo mínimo de terminal no PC2

A programação atual da empresa considera um tempo de terminal no PC de virada (PC2) de 5 minutos. Por este motivo, tal situação foi submetida ao modelo, considerando-se 3 e 5 minutos como tempos mínimos no terminal PC2. Para o

Tabela 7.19: Programações sem troca de linha e tempos mínimos no PC2 de 3 e 5 minutos.

Tempo no PC1	Programação 1 3 minutos		Programação 2 5 minutos	
	hh:mm	variação	hh:mm	variação
TVM	46:05	+ 02:25	46:25	+ 02:45
TET	110:25	− 46:30	111:32	− 45:23
TG	214:47	+ 30:40	216:58	+ 32:51
NRG	31	+ 3	32	+ 4
NV	94	0	94	0
Custo	13.085	− 15,6%	13.222	− 14,7%

tempo mínimo de zero minutos foi encontrado o mesmo resultado da tabela (7.18).

7.4.3 Troca de Linhas no Grupo

A liberdade para que um veículo possa operar em diferentes linhas dentro do seu grupo aumenta as possibilidades de otimização da operação. Neste caso, devido às características da operação, foi permitido que um veículo alternasse sua operação dentro do seu conjunto: entre as linhas Perimetrais ou as linhas Diametrais. Esta

situação é mais flexível do que a anterior e ainda respeita as principais limitações operacionais, ou seja, a coloração do veículo e o tipo da linha. Os resultados abaixo mostram programações cujas reduções no custo operacional são ainda mais significativas do que nos casos anteriores.

7.4.3.1 Tempo mínimo de terminal no PC1

Considerando tempos de terminal no ponto de início da operação de 0, 3 e 5 minutos, foram geradas programações cujas características são apresentadas na tabela (7.20). Observa-se que os percentuais de otimização obtidos neste caso são maiores do que os respectivos percentuais na tabela (7.18).

Tabela 7.20: Programações com troca de veículo entre linhas do mesmo grupo, e tempos mínimos no PC1 de 0, 3 e 5 minutos.

Tempo no PC1	Programação 1 0 minutos		Programação 2 3 minutos		Programação 3 5 minutos	
	hh:mm	variação	hh:mm	variação	hh:mm	variação
TVM	47:10	+ 03:30	48:15	+ 04:35	47:40	+ 04:00
TET	85:02	− 71:53	95:51	− 61:04	99:10	− 57:45
TG	216:32	+ 32:25	219:16	+ 35:09	216:01	+ 31:54
NRG	30	+ 2	32	+ 4	31	+ 3
NV	94	0	94	0	94	0
Custo	11.662	− 24,7%	12.501	− 19,3%	12.600	− 18,7%

7.4.3.2 Tempo mínimo de terminal no PC2

Uma vez que os tempos de terminal foram deslocados para o PC2, o modelo gerou programações cujas características são apresentadas na tabela (7.21). O custo destas programações são menores do que os respectivos custos da tabela (7.20). Isso reflete características na tabela de horário que permitem ligeira redução no custo quando os tempos livres ocorrem no PC2.

Tabela 7.21: Programações com troca de veículo entre linhas do mesmo grupo, e tempos mínimos no PC2 de 3 e 5 minutos.

Tempo no PC1	Programação 1 3 minutos		Programação 2 5 minutos	
	hh:mm	variação	hh:mm	variação
TVM	47:20	+ 03:25	47:05	+ 03:25
TET	95:42	− 59:10	97:45	− 59:10
TG	216:26	+ 31:54	216:01	+ 31:54
NRG	30	+ 2	30	+ 2
NV	94	0	94	0
Custo	12.322	− 20,5%	12.415	− 19,9%

7.4.4 Troca Geral de Linhas

Esta situação tem interesse especulativo, visto que sua implantação é praticamente impossível. Permitir que um veículo opere em qualquer linha de domínio da empresa implica as seguintes complicações operacionais: *i*) determinados motoristas devem conhecer o itinerário de várias linhas, com diferentes características, e *ii*) no caso da cidade de Belo Horizonte, veículos que operam em diferentes tipos de linha possuem colorações distintas.

7.4.4.1 Tempo mínimo de terminal no PC1

Assim como nos casos anteriores, inicialmente foram considerados tempos de terminal de 0, 3 e 5 minutos no PC1, cujos resultados constam na tabela (7.22). Estas são as programações que alcançam o maior grau de otimização, mas também são aquelas que apresentam as maiores dificuldades de implementação, quando possível.

7.4.4.2 Tempo mínimo de terminal no PC2

Para os tempos de terminal ocorrendo no extremo oposto ao ponto de partida, isto é, no PC2, foram obtidos os seguintes resultados:

Embora as programações acima apresentem um grau de otimização bastante superior às programações que permitem a troca de linha no mesmo grupo (seção 7.4.2), a diferença entre elas não atinge 9,5 %.

Tabela 7.22: Programações com troca de veículos entre todas as linha, e tempos mínimos no PC1 de 0, 3 e 5 minutos.

Tempo no PC1	Programação 1 0 minutos		Programação 2 3 minutos		Programação 3 5 minutos	
	hh:mm	variação	hh:mm	variação	hh:mm	variação
TVM	46:25	+ 02:45	47:55	+ 04:15	47:30	+ 03:50
TET	81:37	− 75:18	92:54	− 64:01	95:33	− 61:22
TG	204:48	+ 20:41	211:37	+ 27:30	210:56	+ 26:49
NRG	28	0	29	+ 1	29	+ 1
NV	94	0	94	0	94	0
Custo	11.307	− 27,0%	12.194	− 21,3%	12.303	− 20,6%

Tabela 7.23: Programações com troca de veículos entre todas as linha, e tempos mínimos no PC2 de 3 e 5 minutos.

Tempo no PC1	Programação 1 3 minutos		Programação 2 5 minutos	
	hh:mm	variação	hh:mm	variação
TVM	47:05	+ 03:35	47:15	+ 03:35
TET	92:50	− 62:37	94:18	− 62:37
TG	206:26	+ 24:09	208:16	+ 24:09
NRG	28	0	28	0
NV	94	0	94	0
Custo	12.060	− 22,2%	12.168	− 21,5%

7.4.5 Conclusões da Aplicação em Belo Horizonte

Foram encontradas diferentes programações que reduzem o tempo total de permanência da frota nos terminais. Mesmo considerando a situação menos favorável à otimização, ou seja, aquela que não altera a filosofia operacional da empresa (seção 7.4.2), foram encontradas programações com margem de otimização variando entre 14% e 20%, tendo em vista a função de custo (7.1).

Embora a situação intermediária, que permite a troca dos veículos entre linhas de um mesmo grupo, exija a mudança na forma de operação da empresa, as programações geradas são interessantes, por serem de implantação possível e por apresentarem margens de otimização variando entre 17% e 25% em relação à programação corrente. Tais programações são viáveis na prática, pois elas *i*) restringem a diversidade de linhas nas quais veículo e tripulação operam, e *ii*) respeitam a relação da coloração

dos veículos com as linhas nas quais estes operarão.

Permitir a troca generalizada entre as linhas sob responsabilidade da empresa é uma situação de difícil implementação, e tem com objetivo comparar as programações de ótimo global com aquelas de interesse prático. Nesta situação foram observadas taxas de otimização que variam entre 20% e 27%. Tendo em vista as pequenas diferenças percentuais percebidas entre esta situação e a situação intermediária; e as grandes dificuldades operacionais na viabilização desta situação, pode-se concluir que as soluções descritas na seção (7.4.3) são as mais interessantes do ponto de vista prático.

7.5 Conclusões

Neste trabalho foram exploradas duas metodologias computacional para resolver o problema de programação de veículos com uma única garagem - PPV. A metodologia que formula o problema como um problema de fluxo com custo mínimo e posteriormente faz a eliminação dos arcos longos é a mais eficiente na redução do tamanho da rede. Entretanto, aquela que formula o problema como uma pseudo-designação, e utiliza a técnica de geração do arcos associada ao algoritmo *Out-of-Kilter*, é mais eficiente quanto ao tempo de obtenção da resposta, além de permitir a inclusão de restrições adicionais.

Utilizando a segunda metodologia, foi possível formular e resolver problemas reais de programação de veículos inseridos em diferentes realidades operacionais.

Os casos de *Reading* e Sorocaba serviram para demostrar a eficiência da metodologia comparando seus resultados com os resultados do sistema *BOOST*.

No caso de Santos, foram feitas adaptações para que o modelo gerasse soluções alternativas, relaxando-se a obrigatoriedade do cumprimento de todas as viagens programadas. Diferentes funções de penalisação para a omissão das viagens foram testadas, gerando diversas alternativas para a redução da frota em operação. Uma variante do modelo foi desenvolvida para definir a melhor distribuição de um conjunto de viagens, tendo em vista a taxa de ocupação do veículo e o tempo máximo

de espera do passageiro no terminal. Tal implementação não foi testada pela falta de dados reais.

O problema de Belo Horizonte foi resolvido com diferentes níveis de troca de linha por parte dos veículos em operação e variados tempos de terminal após o cumprimento de cada viagem. Neste caso foi aprimorado o parâmetro "tempo de terminal", que deixou de ser igual para todas as viagens, passando a depender da linha e faixa horária da viagem. Esta flexibilização teve como objetivo gerar programações que também atendessem a determinadas restrições da tripulação.

Estas aplicações confirmaram as expectativas da pesquisa, pois a eficiência do método foi validada frente ao modelo heurístico *BOOST*. Quando aplicada a casos onde havia margem de otimização, o modelo gerou diferentes alternativas operacionais, sempre com diminuição no custo total da programação . No caso em que não havia margem de otimização, foi possível adaptar o modelo para gerar programações com redução da frota pela eliminação das viagens menos importantes ao sistema. As diferentes situações enfrentadas mostram que a metodologia proposta é capaz de resolver o problema proposto, e ainda incorporar variações operacionais presentes nos casos reais.

Capítulo 8

Conclusões e Recomendações

8.1 Introdução

Neste trabalho foram explorados dois métodos de resolução do Problema Básico de Programação de Veículos, que utilizam Algoritmos de Fluxo em Redes. Para resolver os problemas subjacentes, foi empregado o algoritmo primal-dual *Out-of-Kilter*. As principais conclusões obtidas neste trabalho são apresentadas abaixo.

8.2 Conclusões

Dentre os métodos estudados, aquele que utiliza a técnica de eliminação dos arcos longos se mostrou mais eficiente na redução da rede. No entanto, este método requer mais tempo de processamento e a estrutura da sua rede não permite considerar características comumente encontradas na prática.

O segundo método explorado, ou seja, aquele que utiliza a técnica de geração de arcos, embora não tenha sido o mais eficiente na redução da rede, foi o que consumiu menor tempo de processamento. Além disso, sua estrutura permite representar restrições adicionais inerentes aos casos reais abordados. Desta forma, tal método foi adotado nos estudos de casos, e melhorado no sentido de satisfazer a determinados condicionantes operacionais da realidade brasileira.

Com a adaptação da técnica de geração de arcos, foi possível resolver problemas

de grande porte, como nos casos de Sorocaba. No entanto, para problemas de tal magnitude, verificou-se que o método heurístico *BOOST* chega a ser 8 vezes mais rápido do que o método exato. Esta é uma diferença considerável para tempos de processamento muito grandes, porém torna-se irrelevante para tempos menores, que é o casos dos estudos de Sorocaba.

Embora o método heurístico *BOOST* encontre a solução mais rapidamente do que o método exato proposto, nem sempre elas representam o ótimo global. Nos estudos realizados, foram detectados alguns problemas cuja solução ótima não foi encontrada pelo método heurístico. Nestes casos as diferenças são pequenas, mas como as programações são executadas diariamente, elas podem acarretar uma economia significativa a médio e longo prazo.

Portanto, problemas de grande porte e com diferentes características operacionais podem ser resolvidos eficientemente através da técnica de geração de arcos associada ao algoritmo *Out-of-Kilter*. O desempenho da metodologia desenvolvida neste trabalho possibilita resolver problemas reais de empresas de transporte público, levando a uma possível economia nos custos fixos e principalmente nos custos variáveis das empresas que atuam no setor.

8.3 Contribuições

Em linhas gerais, este trabalho vem contribuir com o desenvolvimento e a aplicação de métodos de otimização em sistemas de transportes públicos brasileiros, pois é apresentado um levantamento dos principais problemas que surgem neste contexto e as principais formulações e métodos de resolução.

Uma vez determinada uma estratégia eficiente para solucionar o problema básico, novas restrições foram incorporadas ao modelo e diferentes estudos de caso foram realizados para validá-lo. Com a apresentação e discussão dos resultados obtidos e a descrição das dificuldades enfrentadas, pretende-se orientar futuros trabalhos que venham a ser realizados nesta área.

Especificamente, são apontadas as seguintes contribuições deste trabalho:

a) Análise do Estado da Arte na resolução dos problemas de programação de veículos, apresentando as principais formulações e métodos de resolução.

b) Exploração de duas importantes técnicas de resolução do problema básico de programação de veículos:

- Eliminação dos arcos longos, e
- Geração dos arcos longos.

c) Abordagem do PPV como um Problema de Circulação, resolvendo-o com o algoritmo *Out-of-Kilter* associado à técnica de geração dos arcos longos.

d) Adaptações introduzidas no Problema de Circulação, flexibilizando-o para:

- considerar o problema de retorno à garagem,
- considerar preferências de ligação entre diferentes linhas,
- gerar programações com omissão de viagens, considerando diferentes custos de penalização,
- gerar a tabela de horários integrada à programação dos veículos, e
- gerar programações de veículos prevendo tempos de troca para a tripulação.

e) Realização de diferentes estudos de caso, apresentado as soluções obtidas e levantando os problemas encontrados.

8.4 Dificuldades Encontradas

A maior dificuldade encontrada neste trabalho é a mesma que vigora em diversas aplicações na área de Pesquisa Operacional, ou seja, a obtenção e a confiabilidade dos dados de entrada. A natureza do problema torna tais dificuldades ainda maiores, visto que os principais dados se referem aos tempos de viagem de ônibus entre os terminais. A situação do tráfego nas cidades brasileiras vem se agravando a cada ano, tornando impossível qualquer previsão quanto aos tempos de duração das viagens (Vasconcellos 2000).

Nas cidades de pequeno e médio porte, nas quais os tempo de viagem são bem determinados, não existe margem de otimização devido ao reduzido tamanho do problema, ou pela grande influência das empresas operadoras junto ao poder público. Desta forma, as empresas operadoras têm total controle quanto à definição do quadro de horários a ser executado. Assim, os órgãos gestores atendem muito mais às solicitações das empresas de transporte público do que aos interesses dos usuários do serviço.

Existe um desinteresse e às vezes até resistência por parte das empresas em fornecer os dados necessários para a realização de estudos da programação dos seus veículos. Em parte tal comportamento pode ser justificado pelas seguintes razões:

a) descrença na possibilidade de existir uma solução melhor do que a solução implementada pela empresa. Esta razão está associada ao despreparo técnico ainda vigente no meio.

b) receio de que um estudo mais detalhado seja capaz de detectar irregularidades favoráveis à empresa, tal como folgas remuneradas pelo orgão gestor do transporte público municipal.

c) receio da média administração pela reação dos motoristas e cobradores ao serem impostas operações mais enxutas em relação aos tempos do sistema. Para evitar tais problemas, a área responsável fornece tempos maiores do que aqueles na operação vigente.

Em muitos casos, os dados fornecidos não representam fidedignamente a realidade do dia-a-dia. As tabelas de horários são antigas e os tempos de viagem são cumpridos com dificuldades.

Existe também uma resitência por parte da alta direção em implementar soluções que apontam para uma redução nos custos totais, mas que requer uma mudança na forma de operação do sistema. Tradicionalmente, a empresas brasileiras adotam práticas operacionais com as seguintes características:

a) os veículos são cativos às linhas,

b) os motoristas são responsáveis pelos danos causados aos veículos,

c) os cobradores devem emitir um relatório de controle da catraca ao final de seus turnos de trabalho,

d) os cobradores devem limpar a parte interna do veículo ao final dos seus turnos.

Estas práticas exigem um tempo de terminal relativamente grande quando ocorre a troca da tripulação durante a operação, além de dificultar a implementação de soluções que consideram a possibilidade de remanejamento dos veículos entre diferentes linhas durante a operação.

A mudança nas práticas gerenciais e operacionais vigentes é fundamental para permitir a geração e implementação de soluções que minimizem os custos de manutenção do sistema de transporte público no Brasil.

8.5 Continuidade e Extensões

Devido ao bom desempenho do método adotado, surgem várias perspectivas de continuidade desta pesquisa. Porém, algumas implementações desenvolvidas neste trabalho, principalmente no que se refere à omissão de viagens planejadas e à geração integrada da tabela de horários com a programação dos veículos, devem ser testadas com dados reais e aprimoradas de acordo com a situação.

Os problemas mais complexos que tem o PPV como um subproblema podem ser explorados utilizando o método desenvolvido neste trabalho, pois este se mostrou eficiente na resolução do problema básico. Algumas heurísticas apresentadas neste trabalho podem ser empregadas na resolução dos problemas mais complexos freqüentemente encontrados, como o programação de veículos com várias garagens e com diferentes tipos de veículos.

Uma vez que os algoritmos de fluxo em redes são capazes de resolver problemas muito grandes em um razoável tempo de processamento, e como existem proposições de modelos de fluxo em redes para resolver os problemas subsequentes de Programação da Tripulação e seu rodízio (Carraresi e Gallo 1984), torna-se natural a continuidade

desta pesquisa neste caminho.

Apêndice A

Programação Linear

Neste capítulo será apresentada a notação básica utilizada no texto, assim como conceitos de programação linear, de teoria dos grafos e de fluxo em redes indispensáveis para o entendimento das técnicas abordadas neste trabalho. Os leitores habituados a tais conceitos podem seguir a leitura para o capítulo seguinte. A notação adotada assim como a descrição de métodos matemáticos empregados neste trabalho está baseada nas seguintes referências: *i*)Geração de Colunas - Chvátal (1980) e Lasdon (1970), e *ii*) Relaxação Lagrangeana - Fisher (1985).

A.1 Programação Linear e Dualidade

Dada uma matriz $A_{m\times n}$, um vetor $b_{m\times 1} \geq 0$ e um vetor $c_{1\times n}$, um problema de programação linear (PPL) consiste em encontrar um vetor $x^*_{n\times 1}$ em $W = \{x_{n\times 1} | Ax \leq b, x \geq 0\}$ que maximize a função linear $c^T x$. Um problema de programação linear na forma canônica é apresentado como segue:

$$\begin{array}{lll} \text{Max} & c^T x & \\ \text{sujeito a} & Ax & \leq b \\ & x & \geq 0 \end{array} \tag{A.1}$$

Um vetor $x \in W$ é dito *solução factível* do PPL, e uma solução factível x^* é dita *solução ótima* do PPL se e somente se $c^T x^* \geq c^T x,\ \forall x \in W$. A função linear $c^T x$ é dita *função objetivo* do PPL. A mesma terminologia é utilizada para problemas de

minimização, bastando trocar Max por Min e inverter devidamente o conceito de solução ótima.

Este modelo genérico pode ser associando a uma empresa que produzir n itens distintos a partir de m recursos disponíveis. Logo, para $i = 1, 2, ..., m$ e $j = 1, 2, ..., n$ tem-se:

- x_j = o nível de produção do item j,
- c_j = lucro unitário referente ao item j,
- a_{ij} =quantidade do recurso i consumida na produção de um item j, e
- b_i = quantidade do recurso i disponível para os n itens produzidos.

Desta forma, a função objetivo está associada a maximizar o lucro da empresa, enquanto as n restrições garantem que o consumo total dos recursos não ultrapasse às quantidades disponíveis. As restrições $x \geq 0$ indicam a não negatividade dos níveis de produção.

Para todo modelo de programação linear existe um modelo correspondente formado pelos mesmos coeficientes porém dispostos de forma diferente. O modelo original é chamado de *Primal* e o seu correspondente é dito *Dual* do primeiro. Considerando o PPL descrito em (A.1) como o Primal, então o seu Dual é dado por:

$$\begin{array}{lll} \text{Min} & y^T b & \\ \text{sujeito a} & A^T y & \geq c^T \\ & y & \geq 0 \end{array} \tag{A.2}$$

Observando os problemas Primal e Dual definidos acima, pode-se concluir que:

a) o dual de um problema de maximização é um problema de minimização.

b) as m restrições primais $\sum_{j=1}^{n} a_{ij}x_j \leq b_i,\ i = 1, 2, ..., m$ estão em correspondência uma a uma com as m variáveis duais y_i, da mesma forma que as n restrições duais $\sum_{i=1}^{m} a_{ji}y_i \geq c_j, j = 1, 2, ..., n$ estão em correspondência uma a uma com as n variáveis primais x_j.

c) os coeficientes de cada variável na função objetivo, primal ou dual, aparecem no outro problema como o lado direito da restrição correspondente.

Os problemas Primal e Dual estão relacionados pelos seguintes Teoremas:

Teorema 2.1 Se x_1 e y_1 são soluções factíveis para o par de problemas primal e dual então $c^T x_1 \geq y_1$. Se $c^T x_1 = y_1^T b$, então x_1 e y_1 são soluções ótimas de seus respectivos problemas.

Como conseqüência deste teorema é possível derivar as seguintes condições de otimalidade:

Teorema 2.2 Considere os problema primal P e seu respectivo dual D apresentados acima. Se ambos os problemas apresentam soluções factíveis, então existem soluções ótimas x^* e y^* tal que para cada coluna j de A ocorre:

$$x_j^* > 0 \Leftrightarrow y^{*T} A_{.j} = c_j \tag{A.3}$$

$$y^{*T} A_{.j} < c_j \Leftrightarrow x_j^* = 0 \tag{A.4}$$

A.1.1 Significado Econômico do Problema Dual

Considerando a interpretação prática dada anteriormente ao modelo (A.1), pode-se fazer a seguinte análise dimensional das grandezas envolvidas:

$$c_j = \left[\frac{\$}{\text{unidade do item }_j}\right]$$

$$a_{ij} = \left[\frac{\text{unidade do recurso }_i}{\text{unidade do item}_j}\right]$$

Para que ambos os lados das restrições duais tenham a mesma dimensão é necessário que:

$$y_i = \left[\frac{\$}{\text{unidade do recurso }_i}\right]$$

Portanto, a variável y_i do dual representa o valor unitário do recurso i, dentro do âmbito do problema analisado e não o valor de mercado daquele recurso. Na solução ótima, y_i^* representa a taxa de aumento (diminuição) na função objetivo Z^*, se a quantidade disponível do recurso i for aumentada (diminuída) dentro de um

determinado limite. Esse limite corresponde ao intervalo de variação de b_i no qual a solução ótima permanece a mesma. Assim sendo, a variável dual y_i^* é chamada de *valor marginal*, *valor implícito*, *preço sombra*, etc.

Apêndice B

Relaxação Lagrangeana

A técnica de Relaxação Lagrangeana se basea no fato de que vários problemas de programação inteira, considerados difíceis, podem ser modelados como problemas fáceis complicados por um conjunto adicional de restrições.

Assim, o problema Lagrangeano é definido substituindo-se o conjunto de restrições complicadoras no domínio do problema, por uma penalização na função objetivo. Esta penalização é constituída pelo produto entre as quantidades violadas pelas restrições e suas respectivas variáveis duais.

O problema Lagrangeano é de fácel solução e fornece um limitante inferior, para o caso de minimização, do valor ótimo do problema original. Para definir o conceito de Relaxação Lagrangeana, seja o seguinte problema de programação inteira (PPI):

$$\begin{array}{rlll} Z = \text{Min} & c^T x & \text{sujeito a} \\ & Ax \geq b & \\ & Dx \geq e & \\ & x \geq 0 & \text{e inteiro,} \end{array} \tag{B.1}$$

e considere que este problema se torne relativamente simples caso o conjunto de restrições $Ax \geq b$ seja removido do domínio do problema.

Dado um vetor não negativo de multiplicadores $u_{m\times 1} \geq 0$, adiciona-se o termo não positivo $u(b - Ax)$ à função objetivo de (B.1), levando ao seguinte problema:

$$\begin{array}{rlll} Z_D(u) = \text{Min} & c^T x + u(b - Ax) & \text{sujeito a} \\ & Ax \geq b & \\ & Dx \geq e & \\ & x \geq 0 & \text{e inteiro} \end{array} \tag{B.2}$$

Visto que foi adicionado um termo não positivo à função objetivo, a solução deste problema corresponde a um limitante inferior de Z.

Para definir o problema de Lagrange deve-se retirar o conjunto de restrições $Ax \geq b$ do problema acima. A supressão de tais restrições não aumenta o valor de Z_D, logo o problema de Lagrange (LR_u), apresentado abaixo, é um limitante inferior do problema original (B.1).

$$\begin{array}{lll} Z_D(u) = \text{Min} & c^T x + u(b - Ax) & \text{sujeito a} \\ & Dx \geq e & \\ & x \geq 0 & \text{e inteiro} \end{array} \tag{B.3}$$

Surge então o problema em definir o valor do vetor de multiplicadores u. Tal vetor pode ser calculado por um processo geral dito *método do subgradiente*, ou por métodos que tiram proveito da estrutura do problema estudado. A seguir encontra-se uma breve descrição do método do subgradiente, um dos métodos mais utilizados para tal.

B.0.2 Método do Subgradiente para as Variáveis Duais

Teoricamente o vetor de multiplicadores u deveria resolver o seguinte problema dual:

$$Z_D = \text{Max } Z_D(u), \quad u \geq 0 \tag{B.4}$$

A função $Z_D(u)$ é convexa e diferenciável, exceto nos pontos onde o problema Lagrangeano tem soluções ótimas múltiplas. Nos pontos diferenciáveis a derivada de $Z_D(u)$ é dada pela expressão $Ax - b$, onde x é uma solução ótima de (LR_u) para um determinado u. Esta expressão é utilizada para minimizar a função $Z_D(u)$. Nos pontos onde $Z_D(u)$ é não diferenciável, o método do subgradiente escolhe arbitrariamente uma solução ótima do conjunto de soluções alternativas do Lagrangeano, e usa o vetor $Ax - b$ em substituição ao gradiente da função. O resultado é um procedimento que determina, a partir de um ponto inicial u^0, uma seqüência de valores para u dada por:

$$u^{k+1} = \max \{0,\ u^k - t_k(b - Ax^k)\}$$

Nesta fórmula, t_k define o tamanho do passo e x^k é a solução ótima do problema LR_u^k, o problema Lagrangeano com variável dual assumindo valor u^k.

Lasdon (1970) mostra que:

$$\text{se } \lim_{k\to\infty} t_k = 0 \text{ e } \lim_{k\to\infty} \sum_{i=1}^{k} t_i = \infty \text{ então } Z_D(u^k) \to Z_D^*$$

A não diferenciabilidade requer uma adaptação no cálculo do passo t_k. Uma expressão para o cálculo de t_k que tem-se mostrado muito eficiente é:

$$t_k = \frac{\lambda_k(Z_D(u^k) - Z^*)}{\|b - Ax^k\|^2}$$

onde Z^* é o melhor valor ótimo dentre as soluções factíveis conhecidas do problema (B.1) e λ_k é um escalar escolhido entre 0 e 2. Normalmente λ é inicializado com valor 2 e posteriormente dividido por 2 sempre que $Z_D(u^k)$ não sofre melhoria em um determinado número de iterações.

Outros procedimentos usados para definir os multiplicadores são ditos métodos de ajuste dos multiplicadores. Estes métodos são heurísticas para o problema dual, desenvolvidas para uma aplicação específica e que exploram alguma estrutura especial do problema associado à aplicação.

Apêndice C

Exemplo de uma Aplicação

Neste apêndice são apresentados os dados de entrada e a programação ótima para duas linhas perimetrais da cidade de Belo Horizonte. Como tais linhas pertencem aos mesmo grupo, e portanto apresentam seus veículos com a mesma coloração, foi permitida a troca de linha durante a operação, portanto os veículos podem atuar nas linha durante o período da sua operação.

C.1 Dados de Entrada

Abaixo estão listados os parâmetros que determinam as características da programação resultante.

Tabela C.1: Parâmetros do Modelo

Tempo Mínimo de Garagem em minutos	60
Peso referente ao custo das viagens mortas (K_1)	2
Peso referente ao do veículo parado no terminal (K_2)	1
Parâmetro de definição de arco curto (em minutos)	60

Cada uma das linhas foi dividas em duas, assim a sub-linha 2152.0 corresponde às viagens do BHShopping para o Shopping Del Rey e a sub-linha 2152.1 do Shopping Del Rey para o BH Shopping.

Para evitar deslocamentos indesejáveis entre dois terminais, estando o veículo fora

Tabela C.2: Dados de duas linhas perimetrias.

Número da Linha	Ponto de controle 1	Ponto de controle 2	Total de veículos	Total de viagens
2152.00 2152.10	Salgado Filho Cruzeiro	Cruzeiro Salgado Filho	7	98
4150.00 4150.10	Shopping Del Rey BH Shopping	BH Shopping Shopping Del Rey	10	128

Tabela C.3: Matriz de Viagens Ociosas

	Garagem	SalFilho	Cruzeiro	ScDelRey	BHShoppi
Garagem	00:00	00:10	00:30	00:10	00:40
SalFilho	00:10	00:00	00:40	00:50	00:20
Cruzeiro	00:30	00:40	00:00	05:00	00:15
ScDelRey	00:10	00:50	05:00	00:00	05:00
BHShoppi	00:40	00:20	00:15	05:00	00:00

de operação, são atribuídos valores elevados os tempos de viagem ociosa entre os respectivos terminais. Neste caso foi adotado o parâmetro de 5 horas como distância entre dois terminais deste tipo, como por exemplo entre os terminais Cruzeiro e Shopping Del Rey.

Tabela C.4: Tabela de Viagens

Vg.	Partida	Loc. Partida	Chegada	Loc. Chegada	Linha
0	04:10	ScDelRey	05:05	BHShoppi	4150.10
1	04:40	ScDelRey	05:35	BHShoppi	4150.10
2	05:00	ScDelRey	05:55	BHShoppi	4150.10
3	05:10	BHShoppi	06:05	ScDelRey	4150.00
4	05:15	ScDelRey	06:10	BHShoppi	4150.10
5	05:30	ScDelRey	06:23	BHShoppi	4150.10
6	05:40	BHShoppi	06:35	ScDelRey	4150.00
7	05:40	ScDelRey	06:35	BHShoppi	4150.10
8	05:50	ScDelRey	06:45	BHShoppi	4150.10
9	06:00	BHShoppi	06:55	ScDelRey	4150.00
10	06:00	SalFilho	06:40	Cruzeiro	2152.00
11	06:00	ScDelRey	07:00	BHShoppi	4150.10
12	06:10	ScDelRey	07:15	BHShoppi	4150.10
13	06:15	BHShoppi	07:10	ScDelRey	4150.00
14	06:15	SalFilho	06:55	Cruzeiro	2152.00
15	06:20	ScDelRey	07:25	BHShoppi	4150.10
16	06:28	BHShoppi	07:21	ScDelRey	4150.00
17	06:30	SalFilho	07:15	Cruzeiro	2152.00
18	06:35	ScDelRey	07:40	BHShoppi	4150.10
19	06:40	BHShoppi	07:35	ScDelRey	4150.00
20	06:45	Cruzeiro	07:35	SalFilho	2152.10
21	06:45	SalFilho	07:35	Cruzeiro	2152.00
22	06:50	BHShoppi	07:45	ScDelRey	4150.00
23	06:50	ScDelRey	07:55	BHShoppi	4150.10
24	07:00	Cruzeiro	07:40	SalFilho	2152.10
25	07:00	SalFilho	07:55	Cruzeiro	2152.00
26	07:05	BHShoppi	08:05	ScDelRey	4150.00
27	07:05	ScDelRey	08:10	BHShoppi	4150.10
28	07:15	SalFilho	08:10	Cruzeiro	2152.00
29	07:20	BHShoppi	08:25	ScDelRey	4150.00
30	07:20	Cruzeiro	08:05	SalFilho	2152.10
31	07:20	ScDelRey	08:25	BHShoppi	4150.10
32	07:30	SalFilho	08:25	Cruzeiro	2152.00
33	07:30	BHShoppi	08:35	ScDelRey	4150.00
34	07:40	Cruzeiro	08:30	SalFilho	2152.10
35	07:40	ScDelRey	08:45	BHShoppi	4150.10
36	07:45	BHShoppi	08:50	ScDelRey	4150.00
37	07:45	SalFilho	08:40	Cruzeiro	2152.00
38	08:00	Cruzeiro	08:55	SalFilho	2152.10
39	08:00	SalFilho	08:55	Cruzeiro	2152.00
40	08:00	BHShoppi	09:05	ScDelRey	4150.00
41	08:00	ScDelRey	09:05	BHShoppi	4150.10
42	08:15	Cruzeiro	09:10	SalFilho	2152.10
43	08:15	BHShoppi	09:20	ScDelRey	4150.00

Continua na próxima página.

Tabela C.4: Continuação

Vg.	Partida	Loc. Partida	Chegada	Loc. Chegada	Linha
44	08:20	SalFilho	09:15	Cruzeiro	2152.00
45	08:20	ScDelRey	09:25	BHShoppi	4150.10
46	08:30	BHShoppi	09:35	ScDelRey	4150.00
47	08:30	Cruzeiro	09:25	SalFilho	2152.10
48	08:40	SalFilho	09:35	Cruzeiro	2152.00
49	08:40	ScDelRey	09:45	BHShoppi	4150.10
50	08:45	Cruzeiro	09:40	SalFilho	2152.10
51	08:50	BHShoppi	09:55	ScDelRey	4150.00
52	09:00	Cruzeiro	09:55	SalFilho	2152.10
53	09:00	SalFilho	09:55	Cruzeiro	2152.00
54	09:00	ScDelRey	10:05	BHShoppi	4150.10
55	09:10	BHShoppi	10:15	ScDelRey	4150.00
56	09:20	Cruzeiro	10:15	SalFilho	2152.10
57	09:20	SalFilho	10:15	Cruzeiro	2152.00
58	09:20	ScDelRey	10:25	BHShoppi	4150.10
59	09:30	BHShoppi	10:35	ScDelRey	4150.00
60	09:40	Cruzeiro	10:35	SalFilho	2152.10
61	09:40	SalFilho	10:35	Cruzeiro	2152.00
62	09:40	ScDelRey	10:45	BHShoppi	4150.10
63	09:50	BHShoppi	10:55	ScDelRey	4150.00
64	10:00	Cruzeiro	10:55	SalFilho	2152.10
65	10:00	SalFilho	10:55	Cruzeiro	2152.00
66	10:00	ScDelRey	11:05	BHShoppi	4150.10
67	10:10	BHShoppi	11:15	ScDelRey	4150.00
68	10:20	Cruzeiro	11:15	SalFilho	2152.10
69	10:20	SalFilho	11:15	Cruzeiro	2152.00
70	10:20	ScDelRey	11:25	BHShoppi	4150.10
71	10:30	BHShoppi	11:35	ScDelRey	4150.00
72	10:40	Cruzeiro	11:35	SalFilho	2152.10
73	10:40	SalFilho	11:35	Cruzeiro	2152.00
74	10:40	ScDelRey	11:45	BHShoppi	4150.10
75	10:50	BHShoppi	11:55	ScDelRey	4150.00
76	11:00	Cruzeiro	11:55	SalFilho	2152.10
77	11:00	SalFilho	11:55	Cruzeiro	2152.00
78	11:00	ScDelRey	12:05	BHShoppi	4150.10
79	11:10	BHShoppi	12:15	ScDelRey	4150.00
80	11:20	Cruzeiro	12:15	SalFilho	2152.10
81	11:20	SalFilho	12:15	Cruzeiro	2152.00
82	11:20	ScDelRey	12:25	BHShoppi	4150.10
83	11:30	BHShoppi	12:35	ScDelRey	4150.00
84	11:35	ScDelRey	12:45	BHShoppi	4150.10
85	11:40	SalFilho	12:35	Cruzeiro	2152.00
86	11:40	Cruzeiro	12:35	SalFilho	2152.10
87	11:50	BHShoppi	12:55	ScDelRey	4150.00
88	11:50	ScDelRey	13:05	BHShoppi	4150.10

Continua na próxima página.

Tabela C.4: Continuação

Vg.	Partida	Loc. Partida	Chegada	Loc. Chegada	Linha
89	12:00	SalFilho	12:55	Cruzeiro	2152.00
90	12:00	Cruzeiro	12:55	SalFilho	2152.10
91	12:10	BHShoppi	13:15	ScDelRey	4150.00
92	12:10	ScDelRey	13:25	BHShoppi	4150.10
93	12:20	SalFilho	13:15	Cruzeiro	2152.00
94	12:20	Cruzeiro	13:15	SalFilho	2152.10
95	12:30	BHShoppi	13:35	ScDelRey	4150.00
96	12:30	ScDelRey	13:40	BHShoppi	4150.10
97	12:40	SalFilho	13:35	Cruzeiro	2152.00
98	12:40	Cruzeiro	13:35	SalFilho	2152.10
99	12:50	BHShoppi	14:00	ScDelRey	4150.00
100	12:50	ScDelRey	14:00	BHShoppi	4150.10
101	13:00	SalFilho	13:55	Cruzeiro	2152.00
102	13:00	Cruzeiro	13:55	SalFilho	2152.10
103	13:10	BHShoppi	14:25	ScDelRey	4150.00
104	13:10	ScDelRey	14:20	BHShoppi	4150.10
105	13:20	SalFilho	14:15	Cruzeiro	2152.00
106	13:20	Cruzeiro	14:15	SalFilho	2152.10
107	13:30	BHShoppi	14:45	ScDelRey	4150.00
108	13:30	ScDelRey	14:40	BHShoppi	4150.10
109	13:40	SalFilho	14:35	Cruzeiro	2152.00
110	13:40	Cruzeiro	14:35	SalFilho	2152.10
111	13:45	BHShoppi	14:55	ScDelRey	4150.00
112	13:50	ScDelRey	15:00	BHShoppi	4150.10
113	14:00	SalFilho	14:55	Cruzeiro	2152.00
114	14:00	Cruzeiro	14:55	SalFilho	2152.10
115	14:05	BHShoppi	15:15	ScDelRey	4150.00
116	14:10	ScDelRey	15:20	BHShoppi	4150.10
117	14:20	SalFilho	15:15	Cruzeiro	2152.00
118	14:20	Cruzeiro	15:15	SalFilho	2152.10
119	14:25	BHShoppi	15:35	ScDelRey	4150.00
120	14:30	ScDelRey	15:40	BHShoppi	4150.10
121	14:40	SalFilho	15:35	Cruzeiro	2152.00
122	14:40	Cruzeiro	15:35	SalFilho	2152.10
123	14:45	BHShoppi	15:55	ScDelRey	4150.00
124	14:50	ScDelRey	16:00	BHShoppi	4150.10
125	15:00	SalFilho	15:55	Cruzeiro	2152.00
126	15:00	Cruzeiro	15:55	SalFilho	2152.10
127	15:05	BHShoppi	16:15	ScDelRey	4150.00
128	15:10	ScDelRey	16:20	BHShoppi	4150.10
129	15:20	SalFilho	16:15	Cruzeiro	2152.00
130	15:20	Cruzeiro	16:15	SalFilho	2152.10
131	15:25	BHShoppi	16:35	ScDelRey	4150.00
132	15:30	ScDelRey	16:40	BHShoppi	4150.10
133	15:40	SalFilho	16:35	Cruzeiro	2152.00

Continua na próxima página.

Tabela C.4: Continuação

Vg.	Partida	Loc. Partida	Chegada	Loc. Chegada	Linha
134	15:40	Cruzeiro	16:35	SalFilho	2152.10
135	15:45	BHShoppi	16:55	ScDelRey	4150.00
136	15:45	ScDelRey	16:55	BHShoppi	4150.10
137	16:00	SalFilho	16:55	Cruzeiro	2152.00
138	16:00	Cruzeiro	16:55	SalFilho	2152.10
139	16:00	ScDelRey	17:10	BHShoppi	4150.10
140	16:05	BHShoppi	17:15	ScDelRey	4150.00
141	16:20	SalFilho	17:15	Cruzeiro	2152.00
142	16:20	Cruzeiro	17:15	SalFilho	2152.10
143	16:20	ScDelRey	17:30	BHShoppi	4150.10
144	16:25	BHShoppi	17:35	ScDelRey	4150.00
145	16:35	ScDelRey	17:45	BHShoppi	4150.10
146	16:40	Cruzeiro	17:35	SalFilho	2152.10
147	16:40	SalFilho	17:35	Cruzeiro	2152.00
148	16:45	BHShoppi	17:55	ScDelRey	4150.00
149	16:50	ScDelRey	18:00	BHShoppi	4150.10
150	17:00	BHShoppi	18:10	ScDelRey	4150.00
151	17:00	Cruzeiro	17:55	SalFilho	2152.10
152	17:00	SalFilho	17:55	Cruzeiro	2152.00
153	17:04	ScDelRey	18:15	BHShoppi	4150.10
154	17:15	BHShoppi	18:25	ScDelRey	4150.00
155	17:15	SalFilho	18:10	Cruzeiro	2152.00
156	17:18	ScDelRey	18:30	BHShoppi	4150.10
157	17:20	Cruzeiro	18:15	SalFilho	2152.10
158	17:30	SalFilho	18:25	Cruzeiro	2152.00
159	17:32	ScDelRey	18:45	BHShoppi	4150.10
160	17:35	BHShoppi	18:45	ScDelRey	4150.00
161	17:40	Cruzeiro	18:35	SalFilho	2152.10
162	17:45	SalFilho	18:40	Cruzeiro	2152.00
163	17:47	ScDelRey	18:55	BHShoppi	4150.10
164	17:50	BHShoppi	19:00	ScDelRey	4150.00
165	18:00	Cruzeiro	18:55	SalFilho	2152.10
166	18:00	ScDelRey	19:10	BHShoppi	4150.10
167	18:05	BHShoppi	19:15	ScDelRey	4150.00
168	18:05	SalFilho	19:00	Cruzeiro	2152.00
169	18:15	Cruzeiro	19:10	SalFilho	2152.10
170	18:15	ScDelRey	19:25	BHShoppi	4150.10
171	18:20	BHShoppi	19:31	ScDelRey	4150.00
172	18:25	SalFilho	19:20	Cruzeiro	2152.00
173	18:30	Cruzeiro	19:25	SalFilho	2152.10
174	18:30	ScDelRey	19:40	BHShoppi	4150.10
175	18:35	BHShoppi	19:47	ScDelRey	4150.00
176	18:45	Cruzeiro	19:40	SalFilho	2152.10
177	18:50	BHShoppi	20:03	ScDelRey	4150.00
178	18:50	SalFilho	19:40	Cruzeiro	2152.00

Continua na próxima página.

Tabela C.4: Continuação

Vg.	Partida	Loc. Partida	Chegada	Loc. Chegada	Linha
179	18:50	ScDelRey	19:55	BHShoppi	4150.10
180	19:00	BHShoppi	20:08	ScDelRey	4150.00
181	19:05	Cruzeiro	20:00	SalFilho	2152.10
182	19:10	ScDelRey	20:15	BHShoppi	4150.10
183	19:15	BHShoppi	20:25	ScDelRey	4150.00
184	19:20	SalFilho	20:10	Cruzeiro	2152.00
185	19:25	Cruzeiro	20:20	SalFilho	2152.10
186	19:30	BHShoppi	20:40	ScDelRey	4150.00
187	19:30	ScDelRey	20:35	BHShoppi	4150.10
188	19:45	Cruzeiro	20:35	SalFilho	2152.10
189	19:45	BHShoppi	20:55	ScDelRey	4150.00
190	19:50	SalFilho	20:40	Cruzeiro	2152.00
191	19:50	ScDelRey	21:00	BHShoppi	4150.10
192	20:00	BHShoppi	21:05	ScDelRey	4150.00
193	20:15	Cruzeiro	21:00	SalFilho	2152.10
194	20:20	BHShoppi	21:25	ScDelRey	4150.00
195	20:20	ScDelRey	21:25	BHShoppi	4150.10
196	20:25	SalFilho	21:10	Cruzeiro	2152.00
197	20:40	BHShoppi	21:45	ScDelRey	4150.00
198	20:45	Cruzeiro	21:30	SalFilho	2152.10
199	20:45	ScDelRey	21:45	BHShoppi	4150.10
200	21:00	SalFilho	21:45	Cruzeiro	2152.00
201	21:05	BHShoppi	22:15	ScDelRey	4150.00
202	21:10	ScDelRey	22:15	BHShoppi	4150.10
203	21:15	Cruzeiro	22:00	SalFilho	2152.10
204	21:30	SalFilho	22:15	Cruzeiro	2152.00
205	21:30	BHShoppi	22:35	ScDelRey	4150.00
206	21:30	ScDelRey	22:35	BHShoppi	4150.10
207	21:50	BHShoppi	22:50	ScDelRey	4150.00
208	21:50	Cruzeiro	22:35	SalFilho	2152.10
209	21:50	ScDelRey	22:55	BHShoppi	4150.10
210	22:00	SalFilho	22:45	Cruzeiro	2152.00
211	22:10	ScDelRey	23:15	BHShoppi	4150.10
212	22:20	BHShoppi	23:20	ScDelRey	4150.00
213	22:20	Cruzeiro	23:05	SalFilho	2152.10
214	22:35	SalFilho	23:15	Cruzeiro	2152.00
215	22:40	BHShoppi	23:40	ScDelRey	4150.00
216	22:40	ScDelRey	23:45	BHShoppi	4150.10
217	22:50	Cruzeiro	23:35	SalFilho	2152.10
218	23:00	BHShoppi	24:00	ScDelRey	4150.00
219	23:10	ScDelRey	24:20	BHShoppi	4150.10
220	23:20	BHShoppi	24:15	ScDelRey	4150.00
221	23:20	Cruzeiro	24:00	SalFilho	2152.10
222	23:50	BHShoppi	24:45	ScDelRey	4150.00
223	24:20	ScDelRey	25:10	BHShoppi	4150.10

Continua na próxima página.

Tabela C.4: Continuação

Vg.	Partida	Loc. Partida	Chegada	Loc. Chegada	Linha
224	24:25	BHShoppi	25:15	ScDelRey	4150.00
225	25:15	BHShoppi	26:05	ScDelRey	4150.00

C.2 Programação Ótima

Abaixo tem-se um resumo das características da programação ótima, seguida pelos blocos das viagens a serem executadas pelos respectivos veículos.

Tabela C.5: Características da Programação Ótima

Número mínimo de veículos	17
Tempo total de viagens vazias	07:00 hh:mm
Tempo total parado no terminal	21:29 hh:mm
Tempo total parado na Garagem	28:06 hh:mm
Numero de retornos à Garagem	4

Tabela C.6: Bloco do Veículo 1

Viagem	Partida		Chegada		TVM	TT	Linha
1	ScDelRey	04:10	BHShoppi	05:05	10	0	4150.00
2	BHShoppi	05:10	ScDelRey	06:05	0	5	4150.10
3	ScDelRey	06:10	BHShoppi	07:15	0	5	4150.00
4	BHShoppi	07:20	ScDelRey	08:25	0	5	4150.10
Retorna à Garagem por 03:05 hs					10	0	
5	ScDelRey	11:50	BHShoppi	13:05	10	0	4150.00
6	BHShoppi	13:10	ScDelRey	14:25	0	5	4150.10
7	ScDelRey	14:30	BHShoppi	15:40	0	5	4150.00
8	BHShoppi	15:45	ScDelRey	16:55	0	5	4150.10
9	ScDelRey	17:04	BHShoppi	18:15	0	9	4150.00
10	BHShoppi	18:20	ScDelRey	19:31	0	5	4150.10

Tabela C.7: Bloco do Veículo 2

Viagem	Partida		Chegada		TVM	TT	Linha
1	ScDelRey	04:40	BHShoppi	05:35	10	0	4150.00
2	BHShoppi	05:40	ScDelRey	06:35	0	5	4150.10
3	ScDelRey	06:35	BHShoppi	07:40	0	0	4150.00
4	BHShoppi	07:45	ScDelRey	08:50	0	5	4150.10
5	ScDelRey	09:00	BHShoppi	10:05	0	10	4150.00
6	BHShoppi	10:10	ScDelRey	11:15	0	5	4150.10
7	ScDelRey	11:20	BHShoppi	12:25	0	5	4150.00
8	BHShoppi	12:30	ScDelRey	13:35	0	5	4150.10
9	ScDelRey	13:50	BHShoppi	15:00	0	15	4150.00
10	BHShoppi	15:05	ScDelRey	16:15	0	5	4150.10
11	ScDelRey	16:20	BHShoppi	17:30	0	5	4150.00
12	BHShoppi	17:35	ScDelRey	18:45	0	5	4150.10
13	ScDelRey	18:50	BHShoppi	19:55	0	5	4150.00
14	BHShoppi	20:00	ScDelRey	21:05	0	5	4150.10
15	ScDelRey	21:10	BHShoppi	22:15	0	5	4150.00
16	BHShoppi	22:20	ScDelRey	23:20	0	5	4150.10

Tabela C.8: Bloco do Veículo 3

Viagem	Partida		Chegada		TVM	TT	Linha
1	ScDelRey	05:00	BHShoppi	05:55	10	0	4150.00
2	BHShoppi	06:00	ScDelRey	06:55	0	5	4150.10
3	ScDelRey	07:05	BHShoppi	08:10	0	10	4150.00
4	BHShoppi	08:15	ScDelRey	09:20	0	5	4150.10
5	ScDelRey	09:20	BHShoppi	10:25	0	0	4150.00
6	BHShoppi	10:30	ScDelRey	11:35	0	5	4150.10
7	ScDelRey	11:35	BHShoppi	12:45	0	0	4150.00
8	BHShoppi	12:50	ScDelRey	14:00	0	5	4150.10
9	ScDelRey	14:10	BHShoppi	15:20	0	10	4150.00
10	BHShoppi	15:25	ScDelRey	16:35	0	5	4150.10
11	ScDelRey	16:35	BHShoppi	17:45	0	0	4150.00
12	BHShoppi	17:50	ScDelRey	19:00	0	5	4150.10
13	ScDelRey	19:10	BHShoppi	20:15	0	10	4150.00
14	BHShoppi	20:20	ScDelRey	21:25	0	5	4150.10
15	ScDelRey	22:10	BHShoppi	23:15	0	45	4150.00
16	BHShoppi	23:20	ScDelRey	00:15	0	5	4150.10
17	ScDelRey	00:20	BHShoppi	01:10	0	5	4150.00
18	BHShoppi	01:15	ScDelRey	02:05	0	5	4150.10

Tabela C.9: Bloco do Veículo 4

Viagem	Partida		Chegada		TVM	TT	Linha
1	ScDelRey	05:15	BHShoppi	06:10	10	0	4150.00
2	BHShoppi	06:15	ScDelRey	07:10	0	5	4150.10
3	ScDelRey	07:20	BHShoppi	08:25	0	10	4150.00
4	BHShoppi	08:30	ScDelRey	09:35	0	5	4150.10
5	ScDelRey	09:40	BHShoppi	10:45	0	5	4150.00
6	BHShoppi	10:50	ScDelRey	11:55	0	5	4150.10
7	ScDelRey	12:10	BHShoppi	13:25	0	15	4150.00
8	BHShoppi	13:30	ScDelRey	14:45	0	5	4150.10
9	ScDelRey	14:50	BHShoppi	16:00	0	5	4150.00
10	BHShoppi	16:05	ScDelRey	17:15	0	5	4150.10
11	ScDelRey	17:18	BHShoppi	18:30	0	3	4150.00
12	BHShoppi	18:35	ScDelRey	19:47	0	5	4150.10
13	ScDelRey	19:50	BHShoppi	21:00	0	3	4150.00
14	BHShoppi	21:05	ScDelRey	22:15	0	5	4150.10

Tabela C.10: Bloco do Veículo 5

Viagem	Partida		Chegada		TVM	TT	Linha
1	ScDelRey	05:30	BHShoppi	06:23	10	0	4150.00
2	BHShoppi	06:28	ScDelRey	07:21	0	5	4150.10
Retorna à Garagem por 09:49 hs					10	0	
3	SalFilho	17:30	Cruzeiro	18:25	10	0	2152.00
4	Cruzeiro	18:30	SalFilho	19:25	0	5	2152.10

Tabela C.11: Bloco do Veículo 6

Viagem	Partida		Chegada		TVM	TT	Linha
1	ScDelRey	05:40	BHShoppi	06:35	10	0	4150.00
2	BHShoppi	06:40	ScDelRey	07:35	0	5	4150.10
3	ScDelRey	07:40	BHShoppi	08:45	0	5	4150.00
4	BHShoppi	08:50	ScDelRey	09:55	0	5	4150.10
5	ScDelRey	10:00	BHShoppi	11:05	0	5	4150.00
6	BHShoppi	11:10	ScDelRey	12:15	0	5	4150.10
7	ScDelRey	12:30	BHShoppi	13:40	0	15	4150.00
8	BHShoppi	13:45	ScDelRey	14:55	0	5	4150.10
9	ScDelRey	15:10	BHShoppi	16:20	0	15	4150.00
10	BHShoppi	16:25	ScDelRey	17:35	0	5	4150.10
11	ScDelRey	17:47	BHShoppi	18:55	0	12	4150.00
12	BHShoppi	19:00	ScDelRey	20:08	0	5	4150.10
13	ScDelRey	20:20	BHShoppi	21:25	0	12	4150.00
14	BHShoppi	21:30	ScDelRey	22:35	0	5	4150.10
15	ScDelRey	22:40	BHShoppi	23:45	0	5	4150.00
16	BHShoppi	23:50	ScDelRey	00:45	0	5	4150.10

Tabela C.12: Bloco do Veículo 7

Viagem	Partida		Chegada		TVM	TT	Linha
1	ScDelRey	05:50	BHShoppi	06:45	10	0	4150.00
2	BHShoppi	06:50	ScDelRey	07:45	0	5	4150.10
3	ScDelRey	08:00	BHShoppi	09:05	0	15	4150.00
4	BHShoppi	09:10	ScDelRey	10:15	0	5	4150.10
5	ScDelRey	10:20	BHShoppi	11:25	0	5	4150.00
6	BHShoppi	11:30	ScDelRey	12:35	0	5	4150.10
7	ScDelRey	12:50	BHShoppi	14:00	0	15	4150.00
8	BHShoppi	14:05	ScDelRey	15:15	0	5	4150.10
9	ScDelRey	15:30	BHShoppi	16:40	0	15	4150.00
10	BHShoppi	16:45	ScDelRey	17:55	0	5	4150.10
11	ScDelRey	18:00	BHShoppi	19:10	0	5	4150.00
12	BHShoppi	19:15	ScDelRey	20:25	0	5	4150.10

Tabela C.13: Bloco do Veículo 8

Viagem	Partida		Chegada		TVM	TT	Linha
1	SalFilho	06:00	Cruzeiro	06:40	10	0	2152.00
2	Cruzeiro	06:45	SalFilho	07:35	0	5	2152.10
3	SalFilho	08:00	Cruzeiro	08:55	0	25	2152.00
4	Cruzeiro	09:00	SalFilho	09:55	0	5	2152.10
5	SalFilho	10:00	Cruzeiro	10:55	0	5	2152.00
6	Cruzeiro	11:00	SalFilho	11:55	0	5	2152.10
7	SalFilho	12:00	Cruzeiro	12:55	0	5	2152.00
8	Cruzeiro	13:00	SalFilho	13:55	0	5	2152.10
9	SalFilho	14:00	Cruzeiro	14:55	0	5	2152.00
10	Cruzeiro	15:00	SalFilho	15:55	0	5	2152.10
11	SalFilho	16:00	Cruzeiro	16:55	0	5	2152.00
12	Cruzeiro	17:00	SalFilho	17:55	0	5	2152.10
13	SalFilho	18:05	Cruzeiro	19:00	0	10	2152.00
14	Cruzeiro	19:05	SalFilho	20:00	0	5	2152.10

Tabela C.14: Bloco do Veículo 9

Viagem	Partida		Chegada		TVM	TT	Linha
1	ScDelRey	06:00	BHShoppi	07:00	10	0	4150.00
2	BHShoppi	07:05	ScDelRey	08:05	0	5	4150.10
3	ScDelRey	08:20	BHShoppi	09:25	0	15	4150.00
4	BHShoppi	09:30	ScDelRey	10:35	0	5	4150.10
5	ScDelRey	10:40	BHShoppi	11:45	0	5	4150.00
6	BHShoppi	11:50	ScDelRey	12:55	0	5	4150.10
7	ScDelRey	13:10	BHShoppi	14:20	0	15	4150.00
8	BHShoppi	14:25	ScDelRey	15:35	0	5	4150.10
9	ScDelRey	15:45	BHShoppi	16:55	0	10	4150.00
10	BHShoppi	17:00	ScDelRey	18:10	0	5	4150.10
11	ScDelRey	18:15	BHShoppi	19:25	0	5	4150.00
12	BHShoppi	19:30	ScDelRey	20:40	0	5	4150.10
13	ScDelRey	20:45	BHShoppi	21:45	0	5	4150.00
14	BHShoppi	21:50	ScDelRey	22:50	0	5	4150.10
15	ScDelRey	23:10	BHShoppi	00:20	0	20	4150.00
16	BHShoppi	00:25	ScDelRey	01:15	0	5	4150.10

Tabela C.15: Bloco do Veículo 10

Viagem	Partida		Chegada		TVM	TT	Linha
1	SalFilho	06:15	Cruzeiro	06:55	10	0	2152.00
2	Cruzeiro	07:00	SalFilho	07:40	0	5	2152.10
3	SalFilho	07:45	Cruzeiro	08:40	0	5	2152.00
4	Cruzeiro	08:45	SalFilho	09:40	0	5	2152.10
5	SalFilho	09:40	Cruzeiro	10:35	0	0	2152.00
6	Cruzeiro	10:40	SalFilho	11:35	0	5	2152.10
7	SalFilho	11:40	Cruzeiro	12:35	0	5	2152.00
8	Cruzeiro	12:40	SalFilho	13:35	0	5	2152.10
9	SalFilho	13:40	Cruzeiro	14:35	0	5	2152.00
10	Cruzeiro	14:40	SalFilho	15:35	0	5	2152.10
11	SalFilho	15:40	Cruzeiro	16:35	0	5	2152.00
12	Cruzeiro	16:40	SalFilho	17:35	0	5	2152.10
13	SalFilho	17:45	Cruzeiro	18:40	0	10	2152.00
14	Cruzeiro	18:45	SalFilho	19:40	0	5	2152.10
15	SalFilho	19:50	Cruzeiro	20:40	0	10	2152.00
16	Cruzeiro	20:45	SalFilho	21:30	0	5	2152.10
17	SalFilho	21:30	Cruzeiro	22:15	0	0	2152.00
18	Cruzeiro	22:20	SalFilho	23:05	0	5	2152.10

Tabela C.16: Bloco do Veículo 11

Viagem	Partida		Chegada		TVM	TT	Linha
1	ScDelRey	06:20	BHShoppi	07:25	10	0	4150.00
2	BHShoppi	07:30	ScDelRey	08:35	0	5	4150.10
3	ScDelRey	08:40	BHShoppi	09:45	0	5	4150.00
4	BHShoppi	09:50	ScDelRey	10:55	0	5	4150.10
5	ScDelRey	11:00	BHShoppi	12:05	0	5	4150.00
6	BHShoppi	12:10	ScDelRey	13:15	0	5	4150.10
7	ScDelRey	13:30	BHShoppi	14:40	0	15	4150.00
8	BHShoppi	14:45	ScDelRey	15:55	0	5	4150.10
9	ScDelRey	16:00	BHShoppi	17:10	0	5	4150.00
10	BHShoppi	17:15	ScDelRey	18:25	0	5	4150.10
11	ScDelRey	18:30	BHShoppi	19:40	0	5	4150.00
12	BHShoppi	19:45	ScDelRey	20:55	0	5	4150.10
13	ScDelRey	21:30	BHShoppi	22:35	0	35	4150.00
14	BHShoppi	22:40	ScDelRey	23:40	0	5	4150.10

Tabela C.17: Bloco do Veículo 12

Viagem	Partida		Chegada		TVM	TT	Linha
1	SalFilho	06:30	Cruzeiro	07:15	10	0	2152.00
2	Cruzeiro	07:20	SalFilho	08:05	0	5	2152.10
3	SalFilho	08:20	Cruzeiro	09:15	0	15	2152.00
4	Cruzeiro	09:20	SalFilho	10:15	0	5	2152.10
5	SalFilho	10:20	Cruzeiro	11:15	0	5	2152.00
6	Cruzeiro	11:20	SalFilho	12:15	0	5	2152.10
7	SalFilho	12:20	Cruzeiro	13:15	0	5	2152.00
8	Cruzeiro	13:20	SalFilho	14:15	0	5	2152.10
9	SalFilho	14:20	Cruzeiro	15:15	0	5	2152.00
10	Cruzeiro	15:20	SalFilho	16:15	0	5	2152.10
11	SalFilho	16:20	Cruzeiro	17:15	0	5	2152.00
12	Cruzeiro	17:20	SalFilho	18:15	0	5	2152.10
13	SalFilho	18:25	Cruzeiro	19:20	0	10	2152.00
14	Cruzeiro	19:25	SalFilho	20:20	0	5	2152.10
15	SalFilho	20:25	Cruzeiro	21:10	0	5	2152.00
16	Cruzeiro	21:15	SalFilho	22:00	0	5	2152.10
17	SalFilho	22:00	Cruzeiro	22:45	0	0	2152.00
18	Cruzeiro	22:50	SalFilho	23:35	0	5	2152.10

Tabela C.18: Bloco do Veículo 13

Viagem	Partida		Chegada		TVM	TT	Linha
1	SalFilho	06:45	Cruzeiro	07:35	10	0	2152.00
2	Cruzeiro	07:40	SalFilho	08:30	0	5	2152.10
3	SalFilho	08:40	Cruzeiro	09:35	0	10	2152.00
4	Cruzeiro	09:40	SalFilho	10:35	0	5	2152.10
5	SalFilho	10:40	Cruzeiro	11:35	0	5	2152.00
6	Cruzeiro	11:40	SalFilho	12:35	0	5	2152.10
7	SalFilho	12:40	Cruzeiro	13:35	0	5	2152.00
8	Cruzeiro	13:40	SalFilho	14:35	0	5	2152.10
9	SalFilho	14:40	Cruzeiro	15:35	0	5	2152.00
10	Cruzeiro	15:40	SalFilho	16:35	0	5	2152.10
11	SalFilho	16:40	Cruzeiro	17:35	0	5	2152.00
12	Cruzeiro	17:40	SalFilho	18:35	0	5	2152.10
13	SalFilho	18:50	Cruzeiro	19:40	0	15	2152.00
14	Cruzeiro	19:45	SalFilho	20:35	0	5	2152.10

Tabela C.19: Bloco do Veículo 14

Viagem	Partida		Chegada		TVM	TT	Linha
1	ScDelRey	06:50	BHShoppi	07:55	10	0	4150.00
2	BHShoppi	08:00	ScDelRey	09:05	0	5	4150.10
Retorna à Garagem por 07:25 hs					10	0	
3	ScDelRey	16:50	BHShoppi	18:00	10	0	4150.00
4	BHShoppi	18:05	ScDelRey	19:15	0	5	4150.10
5	ScDelRey	19:30	BHShoppi	20:35	0	15	4150.00
6	BHShoppi	20:40	ScDelRey	21:45	0	5	4150.10
7	ScDelRey	21:50	BHShoppi	22:55	0	5	4150.00
8	BHShoppi	23:00	ScDelRey	00:00	0	5	4150.10

Tabela C.20: Bloco do Veículo 15

Viagem	Partida		Chegada		TVM	TT	Linha
1	SalFilho	07:00	Cruzeiro	07:55	10	0	2152.00
2	Cruzeiro	08:00	SalFilho	08:55	0	5	2152.10
3	SalFilho	09:00	Cruzeiro	09:55	0	5	2152.00
4	Cruzeiro	10:00	SalFilho	10:55	0	5	2152.10
5	SalFilho	11:00	Cruzeiro	11:55	0	5	2152.00
6	Cruzeiro	12:00	SalFilho	12:55	0	5	2152.10
7	SalFilho	13:00	Cruzeiro	13:55	0	5	2152.00
8	Cruzeiro	14:00	SalFilho	14:55	0	5	2152.10
9	SalFilho	15:00	Cruzeiro	15:55	0	5	2152.00
10	Cruzeiro	16:00	SalFilho	16:55	0	5	2152.10
11	SalFilho	17:00	Cruzeiro	17:55	0	5	2152.00
12	Cruzeiro	18:00	SalFilho	18:55	0	5	2152.10

Tabela C.21: Bloco do Veículo 16

Viagem	Partida		Chegada		TVM	TT	Linha
1	SalFilho	07:15	Cruzeiro	08:10	10	0	2152.00
2	Cruzeiro	08:15	SalFilho	09:10	0	5	2152.10
3	SalFilho	09:20	Cruzeiro	10:15	0	10	2152.00
4	Cruzeiro	10:20	SalFilho	11:15	0	5	2152.10
5	SalFilho	11:20	Cruzeiro	12:15	0	5	2152.00
6	Cruzeiro	12:20	SalFilho	13:15	0	5	2152.10
7	SalFilho	13:20	Cruzeiro	14:15	0	5	2152.00
8	Cruzeiro	14:20	SalFilho	15:15	0	5	2152.10
9	SalFilho	15:20	Cruzeiro	16:15	0	5	2152.00
10	Cruzeiro	16:20	SalFilho	17:15	0	5	2152.10
11	SalFilho	17:15	Cruzeiro	18:10	0	0	2152.00
12	Cruzeiro	18:15	SalFilho	19:10	0	5	2152.10
13	SalFilho	19:20	Cruzeiro	20:10	0	10	2152.00
14	Cruzeiro	20:15	SalFilho	21:00	0	5	2152.10
15	SalFilho	21:00	Cruzeiro	21:45	0	0	2152.00
16	Cruzeiro	21:50	SalFilho	22:35	0	5	2152.10
17	SalFilho	22:35	Cruzeiro	23:15	0	0	2152.00
18	Cruzeiro	23:20	SalFilho	00:00	0	5	2152.10

Tabela C.22: Bloco do Veículo 17

Viagem	Partida		Chegada		TVM	TT	Linha
1	SalFilho	07:30	Cruzeiro	08:25	10	0	2152.00
2	Cruzeiro	08:30	SalFilho	09:25	0	5	2152.10
Retorna à Garagem por 07:47 hs					10	0	
3	ScDelRey	17:32	BHShoppi	18:45	10	0	4150.00
4	BHShoppi	18:50	ScDelRey	20:03	0	5	4150.10

Bibliografia

[1] Ahuja, R. K.; Magnanti, T. L.; Orlin, J. B. (1993) *Network Flows: Theory, Algorithms and Aplications*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

[2] Barnhart, C.; Johnson, E. L.; Nemhauser, G. L.; Savelsbergh, M. W. P.; Vance, P. H. (1998) Branch-and-price: column generation for solving huge integer programs. *Operations Research*, v.46, p.316-329.

[3] Bertossi, A. A.; Carraresi, P.; Gallo, G. (1987) On some matching problems arising in vehicle scheduling models. *Networks*, v.17, p.271-281.

[4] Bertsekas, D. P. (1991) *Linear network optimization: algorithms and codes*. MIT Press, Cambridge, MA.

[5] Bodin, L.; Golden, B.; Assad, A.; Ball, M. (1983) Routing and scheduling of vehicle and crews: The state of the art. *Computers and Operations Research*, v.10, p.63-211.

[6] Bodin, L.; Rosenfield, D.; Kydes, A. (1978) UCOST. A micro approach to a transit planning problem. *Journal of Urban Analysis*, v.5, n.1, p.47-69.

[7] Bokinge, U.; Hasselström, D. (1980) Improved vehicle scheduling in public transport through systematic changes in the time-table. *European Journal of Operations Research*, v.5, p.388-395.

[8] Carraresi, P.; Gallo, G. (1984) Network models for vehicle and crew scheduling. *European Journal of Operational Research*, v.16, p.139-151.

[9] Carpaneto, G.; Dell'Amico, M.; Fischetti, M.; Toth, P. (1989) A branch and bound algorithm for the multiple vehicle scheduling problem. *Networks*, v.19, p.531-548.

[10] Ceder, A. (1995) Minimum cost vehicle scheduling with different types od transit vehicles. In: *Computer-Aided Transit Scheduling*, J. R. Daduna; I. M. Branco; J. M. P. Paixão (eds.), Springer-Verlag, Berlin, p.102-114.

[11] Ceder, A. (1995) Transit vehicle-type scheduling problem. *Transportation Research Record*, v.1503, p.34-38.

[12] Ceder, A.; Stern, H. I. (1981) Deficit function bus scheduling with deadheading trip insertions for fleet size reduction. *Transportation Science*, v.15, p. 338-363.

[13] Chamberlain, M. P.; Wren, A. (1992) Developments and recent experience with the BUSMAN and BUSMAN II system. In: *Computer-Aided Transit Scheduling*, M. Desrochers; J. M. Rousseau (eds.), Springer-Verlag, Berlin, p.1-15.

[14] Cook, S. A. (1971) The complexity of theorem proving procedures. In: *Proceeding of the Third Annual ACM Symposium on the theory of computing*. Association of Computing Machinery, New York, p.151-158.

[15] Costa, A.; Branco, I. M.; Paixão, J. M. P. (1995) Vehicle Scheduling problem with multiple type of vehicles and a single depot. In: *Computer-Aided Transit Scheduling*, J. R. Daduna; I. M. Branco; J. M. P. Paixão (eds.), Springer-Verlag, Berlin, p.115-129.

[16] Crainic, T; Rousseau, J. M. (1987) The column generation principle and the airline crew scheduling problem. *INFOR*, v.25, p.136-151.

[17] Chvátal, V. (1980) *Linear Programming*. W. H. Freeman and Company, New York.

[18] Daduna, J. R.; Mojsilovic, M. (1988) Computer-aided vehicle and duty scheduling using the HOT programme system. In: *Computer-Aided Transit Scheduling*, J. R. Daduna; A. Wren (eds.), Springer-Verlag, Berlin, p. 133-146.

[19] Daduna, J. R.; Paixão, J. M. P. (1995) Vehicle scheduling for public mass transport - an overview. In: *Computer-Aided Transit Scheduling*, J. R. Daduna; I. M. Branco; J. M. P. Paixão (eds.), Springer-Verlag, Berlin, p.76-90.

[20] Daduna, J. R.; Voß, S. (eds.) (2000) Computer-aided Scheduling of Public Transport. Instituts für Wirtschaftswissenschaften, Braunschweig.

[21] Dantzig, G. B.; Fulkerson, D. (1954) Minimizing the number of tankers to meet a fixed scheduling. *Naval Research Logistics Quarterly*, v.1, p.217-222.

[22] Dantzig, G. B.; Fulkerson, D. (1962) *Linear Programming and Extensions*. Princeton University Press, Princeton, New Jersey.

[23] Dantzig, G. B.; Wolfe, P. (1960) Decompositiom principle for linear programs. *Operations Research*, v.8, p.101-111.

[24] Dell'Amico, M.; Fischetti, M.; Toth, P. (1993) Heuristic algorithm for the multiple depot vehicle scheduling problem. *Management Science*, v.39, p.115-125.

[25] Desrochers, M; Soumis, F. (1989) A column generation approach to the urban transit crew scheduling problem. *Transportation Science*, v.23, n.1, p.1-13

[26] Desrosiers, J.; Sauvé, M. , Soumis, F. (1988) Lagrangian relaxation methods for solving the minimum fleet size multiple traveling salesman problem with time windows. *Management Science* v.34, n. 8, p.1005-1022.

[27] Dunlay, W. J. Jr; Gualda, N. D. F. (1979) Flow optimization on a transportation network. *Transportation Engineering Journal of ASCE*, v.105, p.41-57.

[28] El-Azm, A. (1985) The minimum fleet size problem and its aplications to bus scheduling. In: *Computer Scheduling of Public Transport 2*, J. M. Rousseau (ed.), North-Holland, Amsterdam, p.493-512.

[29] Fisher, M. L. (1985) An application oriented guide to Lagrangian relaxation. *Interfaces*, v.15, n.2, p.10-21.

[30] Forbes, M. A.; Holt, J. N.; Watts, A. M. (1994) An exact algorithm for multiple depot bus scheduling. *European Journal of Operational Research*, v.72, p.115-124.

[31] Fores, S. (1996) A column generation approaches to bus driver scheduling. Ph.D. Thesis (Doutorado) - School of Computer Studies, University of Leeds.

[32] Fores, S.; Proll, L.; Wren, A. (1999) An improved ILP system for driver scheduling. In: *Computer-Aided Transit Scheduling*, N. H. M. Wilson (ed.), Springer-Verlag, Berlin, p. 43-61.

[33] Freling, R.; Paixão, J. M. P (1995) Vehicle scheduling with time constraint. In: *Computer-Aided Transit Scheduling*, J. R. Daduna; I. M. Branco; J. M. P. Paixão (eds.), Springer-Verlag, Berlin, p.130-144.

[34] Freling, R.; Paixão, J. M. P.; Wagelmans, A. P. M. (1995) Models and algorithms for vehicle scheduling. Disponível em http://kaa.cs.few.eur.nl/few/people/freling/.

[35] Freling, R.; Wagelmans, A. P. M.; Paixão, J. M. P. (1999) An overview of models and techniques for integrating vehicle and crew scheduling. In: *Computer-Aided Transit Scheduling*, N. H. M. Wilson (ed.), Springer-Verlag, Berlin, p. 441-459.

[36] French, S. (1982) *Sequencing and scheduling*. Ellis Horwood, Chichester.

[37] Fulkerson, D. (1961) An out-of-kilter method for minimal cost flow problems. *SIAM Journal on Applied Mathematics*, v.9, p.18-27.

[38] Gavish, B.; Schweitzer, P.; Shlifer, E. (1978) Assigning buses to schedules in a metropolitan area. *Computers and Operations Research*, v.5, p.129-138.

[39] Gavish, B.; Shlifer, E. (1978) An approach for solving a class of transportation scheduling problems. *Computer and Operations Research*, v.3, p. 122-134.

[40] Hoffstadt, J. (1981) Computerized vehicle and driver scheduling for the Hamburger Hochbahn Akrtieingesellschaft. In: *Computer Scheduling of Public Transport*, Wren A. (ed.), North-Holland, Amsterdam, p.35-52.

[41] Kirkman, F. (1968) Problems of innovation in the transport industry: a bus scheduling program. In: *Proceedings of PTRC Public Transport Analysis Seminar*, Planning and Transport Research and Computation Co. Ltd., London, v.1, p.1-15.

[42] Kuhn, H. W. (1956) Variants of the hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, v.3, p.253-258.

[43] Kwan, R. K.; Rahin, M. A. (1995) Bus scheduling with trip co-ordenation and complex constraints. In: *Computer-Aided Transit Scheduling*, J. R. Daduna; I. M. Branco; J. M. P. Paixão (eds.), Springer-Verlag, Berlin, p.91-101.

[44] Kwan, R. K.; Rahin, M. A. (1999) Object orientede bus vehicle scheduling - the BOOST system. In: *Computer-Aided Transit Scheduling*, N. H. M. Wilson (ed.), Springer-Verlag, Berlin, p. 177-191.

[45] Lamatsch, A. (1992) An approach to vehicle scheduling with depot capacity constraints. In: *Computer-Aided Transit Scheduling*, M. Desrochers; J. M. Rousseau (eds.), Springer-Verlag, Berlin, p.181-195.

[46] Lasdon, S. L. (1970) *Optimization Theory for Large Systems*. Macmillan Publishing Co., New York.

[47] Löbel, A. (1998) Vehicle scheduling in public transit and lagrangean pricing. *Management Science*, v. 44, n. 12, p. 1637-1649.

[48] Löbel, A. (1999) Solving large-scale multiple-depot vehicle scheduling problems. In: *Computer-Aided Transit Scheduling*, N. H. M. Wilson (ed.), Springer-Verlag, Berlin, p. 193-220.

[49] Martin-Löf, A. (1970) A branch-and-bound algorithm for determining the minimal fleet size of a transportatuion system. *Transportation Science*, v.4, p. 159-163.

[50] Manington, P. D.; Wren, A. (1975) Experiences with a bus scheduling algorithm which saves vehicles. In: *Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services*, Chicago.

[51] Mesquita, M.; Paixão, J. M. (1992) Multiple depot vehicle scheduling problem: A new heuristic based on quasi-assignment algorithms. In: *Computer-Aided Transit Scheduling*, M. Desrochers; J. M. Rousseau (eds.), Springer-Verlag, Berlin, p.167-180.

[52] Murty, G. K. (1992) *Network Programming*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

[53] Paixão, J. M. P.; Branco, I. M. (1987) A quasi-assignment algorithm for bus-scheduling. *Networks*, v.17, p.249-269.

[54] Paixão, J. M. P.; Branco, I. M. (1988) Bus scheduling with fixed number of vehicles. In: *Computer-Aided Transit Scheduling*, J. R. Daduna; A. Wren (eds.), Springer-Verlag, Berlin, p. 28-40.

[55] Ribeiro, C.; Soumis, F. (1994) A column generation approach to the multiple depot vehicle scheduling problem. *Operations Research*, v.42, p.41-52.

[56] Saha, J. L. (1970) An algorithm for bus scheduling problems. *Operational Research Quarterly*, v.21, p.463-474.

[57] Schrijver, A. (1989) *Theory of Linear and Integer Programming*. John Wiley & Sons Ltd., Chichester.

[58] Smith, B. M.; Wren, A. (1981) VAMPIRES and TASC: two successfully applied bus scheduling programs. In: *Computer Scheduling of Public Transport*, A. Wren (ed.), North-Holland, Amsterdam, p.97-124.

[59] Stern, H. I.; Ceder, A. (1981) A deficit function approach for bus scheduling. In: *Computer Scheduling of Public Transport*, A. Wren (ed.), North-Holland, Amsterdam, p.85-96.

[60] Vasconcellos, E. A. (2000) *Trasnporte urbano nos países em desenvolvimento: reflexões e propostas*. Annablume, São Paulo.

[61] Wilsom, N. H. M. (ed.) (1999) *Computer-Aided Transit Scheduling*, Springer-Verlag, Berlin.

[62] Wolfenden, K.; Wren, A. (1966) Locomotive scheduling by computer. In: *Proceedings of the British Joint Computer Conference*, IEE Conference publication, Eastbourne, v.19, p.31-37.

[63] Wren, A. (1972) Bus scheduling: and interactive computer method. *Transportation Planning and Technology*, v.1, p.115-122.

[64] Wren, A. (1998) Heuristics ancient and modern: transport scheduling through the ages. *Journal of Heuristics*, v.4, p.87-100.

[65] Wren, A.; Chamberlain, M. P. (1988) The development of Micro-BUSMAN: scheduling on micro-computers. In: *Computer-Aided Transit Scheduling*, J. R. Daduna; A. Wren (eds.), Springer-Verlag, Berlin, p. 160-174.

[66] Wren, A.; Gualda, N. D. F. (1999) Integrated scheduling of buses and drivers. In: *Computer-Aided Transit Scheduling*, N. H. M. Wilson (ed.), Springer-Verlag, Berlin, p. 155-175.

[67] Wren, A.; Kwan, R. S. K. (1999) Installing an urban transport scheduling system. *Journal of Scheduling*, v.2, p.3-17.