Uma abordagem alternativa para os escalonamentos de ônibus e de motoristas

Glauber José Vaz

Dissertação de Mestrado

Instituto de Computação
Universidade Estadual de Campinas

Uma abordagem alternativa para os escalonamentos de ônibus e de motoristas

Glauber José Vaz[1]

Março de 2003

Banca Examinadora:

- Prof. Dr. Arnaldo Vieira Moura (Co-orientador)
- Prof. Dr. Geraldo Robson Mateus
  Departamento de Ciência da Computação - UFMG
- Prof. Dr. Flávio Keidi Miyazawa
  Instituto de Computação - UNICAMP
- Prof. Dr. Ricardo Dahab (Suplente)
  Instituto de Computação - UNICAMP

[1]**Com apoio financeiro da CAPES**

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Vaz, Glauber José

V477a Uma abordagem alternativa para os escalonamentos de ônibus e de motoristas / Glauber José Vaz -- Campinas, [S.P. :s.n.], 2003.

Orientadores : Cid Carvalho de Souza; Arnaldo Vieira Moura

Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Otimização combinatória. 2. Ônibus – Linhas. 3. Programação (Matemática). 4. Transportes coletivos. I. Souza, Cid Carvalho de. II. Moura, Arnaldo Vieira. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

TERMO DE APROVAÇÃO

Tese defendida e aprovada em 28 de março de 2003, pela Banca examinadora composta pelos Professores Doutores:

Prof. Dr. Geraldo Robson Mateus
UFMG

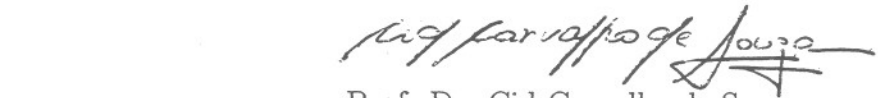
Prof. Dr. Flávio Keidi Miyazawa
IC - UNICAMP

Prof. Dr. Arnaldo Vieira Moura
IC – UNICAMP

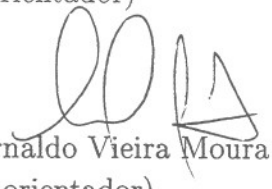
Uma abordagem alternativa para os escalonamentos de ônibus e de motoristas

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Glauber José Vaz e aprovada pela Banca Examinadora.

Campinas, 28 de Março de 2003.



Prof. Dr. Cid Carvalho de Souza
(Orientador)



Prof. Dr. Arnaldo Vieira Moura
(Co-orientador)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Resumo

O planejamento operacional de uma empresa de transporte coletivo envolve os problemas de escalonamento de motoristas, escalonamento de veículos e distribuição dos horários de partida das viagens, todos eles problemas de otimização combinatória. Programação por restrições e programação linear são técnicas empregadas na resolução desses tipos de problemas. Ambas utilizam restrições na modelagem, mas diferem na forma como encontram soluções. Este trabalho trata os problemas relacionados ao planejamento operacional de empresas de ônibus com técnicas híbridas que usam programação por restrições e programação linear.

Abstract

Constructing the operational plan for bus companies raises three main combinatorial optimization problems: vehicle scheduling, driver scheduling and timetabling. Constraint programming and linear programming are combinatorial optimization techniques that can be applied to solve this kind of problems. Both of these techniques use constraints in the model, but they are different in the way they find a solution. This work treats the problem of constructing an operational plan for bus companies using hybrid techniques that exploit the strength of the constraint programming and linear programming techniques when applied to this kind of problems.

Agradecimentos

A Deus pelo que sou e pelas oportunidades profissionais e pessoais com que tem me presenteado.

A meus pais, Valentina e Zezo, pelo amor e apoio que sempre me deram.

À minha irmã, Érika, e à minha namorada, Dany, pelo carinho e pelos 'Te amo's.

A meus orientadores Cid e Arnaldo pela dedicação e por estarem sempre dispostos a ajudar.

A meus familiares, principalmente meus tios José Maria e Lourdes e meus primos Marcelo e Emerson, que me receberam de braços abertos em sua casa na etapa final do meu mestrado, e à Li, que sempre me quebra um galho.

Aos pistolinhas (Baiano, Guido, Triste e Zé) pela amizade sincera e por me terem feito sentir em casa nos vários dias em que fiquei hospedado em sua república.

Ao pessoal do Labinho (Chenca, Ju, Pará e Silvana) e ao William, com quem dividi quitinete, pelo convívio bem-humorado, pacífico e agradável.

À república do Pole (Flavinho, Lásaro, Marcelo) pelas tardes de piscina e bate-papos.

Pelo companheirismo e por tornarem esses anos de mestrado ainda mais compensadores, aos amigos Alessandra, Amanda, Bartho, Bazinho, Borin, Fileto, Flavio, Fred, Joice, Magrão, Maikol, Rogério, Silvânia, Wesley e a galera do futebol coordenada por Marília, Fernando e companhia.

Aos amigos de Uberlândia, que fazem com que meus fins de semana e feriados na cidade sejam sempre descontraídos e alegres.

À CAPES, pelo suporte financeiro.

São tantas as pessoas a quem devo gratidão que, certamente, muitas outras mereciam ser citadas aqui. No entanto, me limitei àquelas com quem mais convivi nestes últimos dois anos e pouco, e que, de certa forma, também foram responsáveis pelo resultado de meu trabalho. Afinal, minha vida profissional sempre foi sustentada por minha vida pessoal.

Cada minuto que passa é uma chance de mudar tudo

Vanilla Sky - filme de Cameron Crowe

Sumário

Lista de Tabelas

Lista de Figuras

Lista de Algoritmos

Capítulo 1

Introdução

Uma vasta quantidade de problemas de otimização pode ser encontrada no planejamento operacional de empresas de transporte coletivo. Um dos problemas mais relevantes e complexos nesta área consiste em realizar a alocação da frota e dos condutores de uma linha de ônibus urbanos. Esse problema foi tratado recentemente em [46]. Resolvê-lo de forma eficiente requer a utilização de algoritmos e técnicas de otimização que produzam uma escala de veículos e de pessoal representando de forma precisa como e quando serão realizadas as viagens em uma certa linha de ônibus, de forma a otimizar certos parâmetros críticos para sua operação. Este planejamento diário é denominado *Problema de Programação de Viagens* e doravante será denotado por PPV.

A dificuldade em realizar este planejamento pode ser percebida quando considerada a grande quantidade de restrições que precisam ser tratadas e os diferentes objetivos a serem atingidos. As restrições, por exemplo, são de vários tipos e incluem desde regras e limites no uso da frota até o cumprimento das leis trabalhistas e acordos sindicais, passando pelas restrições de atendimento à demanda dentro de certos padrões de qualidade de serviço. Quanto aos objetivos, é natural, por exemplo, buscar a minimização do número de veículos e de motoristas para operar a linha e do número de horas extras dos condutores. Contudo, estes objetivos acabam por se tornarem conflitantes, havendo a necessidade, então, de se estabelecer prioridades.

Neste trabalho é estudada a utilização de programação por restrições, programação matemática, heurísticas e a integração destas técnicas para tratar o PPV e seus subproblemas.

1.1 Motivação

O transporte de passageiros é um setor da economia que gera uma grande movimentação de recursos e permite uma contenção de gastos considerável, mesmo quando pequenas

otimizações são feitas. Por isso, o PPV é de extrema importância, já que se trata de um problema real que geralmente é resolvido de forma manual, e baseando-se em conhecimentos práticos adquiridos com a experiência. Por outro lado, o uso de ferramentas computacionais costuma apresentar resultados melhores do que os obtidos manualmente, além de permitir a realização de estudos em menor tempo, com a simulação de diversos cenários e alternativas [32]. Já existem no mercado, alguns *softwares* para esta finalidade, mas as soluções encontradas por eles geralmente acabam por exigir uma quantidade considerável de intervenção humana para tornarem-se operacionais ou, pior ainda, não conseguem chegar a uma redução apreciável de custos. No Brasil, onde os custos de transporte público são altos e as restrições do problema são freqüentemente violadas pelas soluções implementadas nas empresas, existem relativamente poucos estudos nesta área [15, 58, 46, 60].

O PPV ainda é um problema de difícil solução, pois a otimalidade dos resultados não pode ser provada para a grande maioria das instâncias reais. Apesar de a programação matemática ter promovido um grande progresso na resolução dos problemas de escalonamento de tripulação e de veículos, ainda existem restrições difíceis de serem incorporadas em um modelo de programação linear. A programação por restrições é uma técnica que começou a ser empregada na resolução do PPV mais recentemente [8] e ainda há muito trabalho a realizar neste campo.

Além disso, os algoritmos construídos para resolver o PPV, geralmente, baseiam-se em um mesmo modelo. Este trabalho desprende-se de idéias utilizadas anteriormente a fim de produzir maneiras alternativas de se solucionar o problema. Devido à flexibilidade na modelagem e ao controle na busca por soluções, a programação por restrições pode ser uma importante ferramenta nessa busca por algoritmos alternativos.

Muitos problemas de otimização podem ser formulados tanto como problemas de programação linear inteira quanto como problemas de programação por restrições. Enquanto os algoritmos de programação linear inteira são, em geral, bastante eficientes na prova de otimalidade, a programação por restrições pode representar os problemas de uma forma mais compacta e mais natural. Em particular para problemas de escalonamento, os algoritmos de programação linear inteira freqüentemente apresentam grande dificuldade na obtenção de soluções viáveis. Por outro lado, a programação por restrições costuma ser mais eficiente na busca de soluções viáveis para esse tipo de problema [52].

Esse trabalho procura extrair o que há de melhor na programação por restrições e na programação linear, partindo de seus recursos básicos. Ambos os paradigmas suportam desenvolvimento rápido, manutenção econômica e ainda desempenho eficiente na execução dos programas. Mais ainda, a modelagem do problema na forma de restrições resulta em programas simples que podem ser facilmente adaptados, caso haja mudança nos requerimentos do problema. Apesar de tratar modelos parecidos, a programação por restrições

e a programação linear são totalmente diferentes entre si. A possibilidade de uso de duas técnicas distintas aliada às vantagens citadas acima tornam a combinação de programação por restrições com programação linear muito interessante.

Portanto o objetivo é desenvolver um modelo adequado ao cenário brasileiro, diferente dos que normalmente são construídos, utilizando programação linear e programação por restrições, e que encontre uma solução viável para o PPV em tempo razoável.

Assim, esse trabalho pretende contribuir para que empresas, através de um planejamento operacional mais eficiente, possam oferecer transporte público barato, rápido e dentro de um bom padrão de qualidade, respeitando estritamente as restrições trabalhistas e operacionais.

1.2 Descrição do Problema

Há muitos estudos envolvendo os problemas de escalonamento de veículos e de tripulação aplicados aos serviços de transporte rodoviário, aéreo e ferroviário. O presente trabalho considera o escalonamento de ônibus e motoristas no transporte urbano rodoviário.

Conforme [32], os problemas de tomada de decisão das empresas de transporte de passageiros podem ser classificados em estratégicos, táticos e operacionais. A primeira classe diz respeito, por exemplo, à localização de garagens, determinação das regiões que serão atendidas e linhas que serão implantadas. Decisões táticas definem a compra de novos veículos e a definição das políticas de contratação de pessoal. O presente trabalho não se propõe a abordar as decisões táticas e estratégicas das empresas, limitando-se àquelas de cunho operacional. Os aspectos táticos e estratégicos são tidos como entrada no sistema que será implementado, e estão implícitos nos dados de entrada fornecidos. Por exemplo, o fornecimento do tempo de viagem ao longo de uma linha supõe que seus pontos de partida e de chegada já foram definidos.

O planejamento operacional nas empresas de transporte urbano de passageiros compreende as seguintes atividades:

- estudo da demanda;
- montagem das tabelas de horários;
- dimensionamento e alocação da frota;
- dimensionamento e alocação de tripulação.

Os dados referentes à demanda de passageiros serão incluídos como entrada do sistema, que fornecerá soluções viáveis para os outros itens. São problemas típicos do planejamento operacional de uma empresa de transporte coletivo a montagem de horários,

o escalonamento de veículos, o escalonamento de tripulação para curto prazo, eg diário, e o escalonamento de tripulação para longo prazo, eg mensal,[1]. Este documento aborda o conjunto dos três primeiros problemas aplicados a empresas de ônibus urbanos, a que damos o nome de problema de programação de viagens, ou PPV. Portanto, assim como foi caracterizado em [46], encontrar uma solução para o PPV consiste em produzir uma tabela de horários de viagens e escalas de ônibus e funcionários que obedeçam a um conjunto de restrições operacionais e trabalhistas.

Para melhor desenvolvimento deste texto, o escalonamento de tripulação para curto prazo será, doravante, referido por escalonamento de tripulação. Nos trechos em que se queira citar o escalonamento de tripulação para longo prazo, será usado o termo *crew rostering*.

Neste contexto, os únicos funcionários considerados são os motoristas. Em geral, as escalas dos cobradores diferem levemente das escalas dos motoristas, uma vez que um tempo adicional deve ser acrescido às suas jornadas de trabalho de modo a permitir-lhes a entrega da féria na garagem da empresa, ao final de seu turno de trabalho.

Tipicamente, em uma linha de ônibus existem dois pontos que determinam o início e o fim das viagens, a que damos o nome de *pontos de controle* (PC). São nos PCs que os motoristas descansam e aguardam a próxima viagem que terão de fazer. Também são comuns linhas que passam por apenas um PC, denominadas *linhas circulares*. Entretanto, é fácil representá-las como linhas que utilizam dois PCs, para que sejam processadas pelos mesmos algoritmos.

A resposta ao PPV descreve completamente a escala de uso dos veículos e a escala de trabalho dos funcionários. Além disso, apresenta os horários de partida das viagens entre PCs e das viagens que envolvem a garagem e um PC. O principal objetivo é utilizar o mínimo de recursos possível, considerando os custos relativos aos salários dos motoristas, às horas extras e ao uso dos veículos. É essencial também a boa distribuição dos horários de partida das viagens ao longo do dia.

Para gerar a saída do PPV, é necessário que se alimente o sistema com alguns dados de entrada e com as restrições que devem ser satisfeitas. Estes dois aspectos são descritos detalhadamente nas seções 1.2.4 e 1.2.5. Antes, cada sub-problema do PPV é apresentado em maiores detalhes.

1.2.1 Montagem da Tabela de Horários

O comportamento do trânsito nas cidades não é homogêneo durante todo o dia. Há grande oscilação no nível de demanda pelo serviço de transporte público e na velocidade do fluxo

[1] Estes problemas são conhecidos respectivamente por *timetabling*, *vehicle scheduling*, *crew scheduling* e *crew rostering*.

de veículos, por exemplo. Assim, quando se planeja o transporte público, não se pode levar em consideração o dia todo como uma unidade de tempo. É necessário dividi-lo em períodos, em que as características do trânsito sejam mais homogêneas.

Desta forma, antes de montar as tabelas de horários, as empresas de transporte coletivo definem intervalos de tempo para classificar os dados levantados. Normalmente, elas fazem o planejamento operacional baseado em dados que variam para cada faixa horária do dia. Assim, são fornecidas, por exemplo, as demandas de passageiros nos PCs e as durações das viagens de acordo com cada faixa horária.

A fim de tornar o documento mais claro, daqui a diante, o intervalo de tempo considerado será sempre de uma hora. No entanto, na prática, pode haver instâncias do problema com intervalos de tamanhos diferentes de uma hora ou, até mesmo, intervalos de tamanhos diferentes entre si para uma mesma linha de ônibus.

O problema de montar a tabela de horários para um PC de uma determinada linha envolve dois sub-problemas: a definição da quantidade de viagens que devem ser feitas por faixa horária e a atribuição do horário de partida a cada viagem. Enquanto a quantidade de viagens deve ser suficiente para atender à demanda, a distribuição das viagens deve oferecer horários de partida que procuram atender da melhor forma ao usuário, com o objetivo de fazê-lo esperar o mínimo possível no ponto, no pior caso.

1.2.2 Escalonamento de Ônibus

A escala de ônibus corresponde à atribuição de todas as viagens de uma linha aos veículos disponíveis, de modo que cada viagem seja alocada a um único ônibus. Além disso, cada ônibus deve sair da garagem antes da primeira viagem e voltar após a última. O objetivo é minimizar a quantidade de ônibus utilizados, sem contudo, violar as restrições operacionais. Em muitos lugares, um veículo pode ser utilizado em mais de uma linha[2]. No Brasil, esta situação normalmente não ocorre e este trabalho a desconsidera.

Alguns trabalhos prevêem a presença de várias garagens. Entretanto, fixar uma garagem para uma determinada linha simplifica o problema e não afeta a qualidade da solução no caso brasileiro. Também há trabalhos que consideram diferentes tipos de veículos [5, 7], com quantidade de assentos e grau de conforto diferentes, mas no Brasil, não é muito comum esta situação para uma mesma linha. Portanto, a multiplicidade de garagens e de tipos de veículos não faz parte do escopo deste estudo.

[2]O processo de mudar um veículo de uma linha para outra é conhecido por *interlining*.

1.2.3 Escalonamento de Motoristas

A escala de motoristas corresponde à atribuição de todas as viagens de uma linha a motoristas de modo que cada viagem seja alocada a um único motorista. O objetivo é minimizar a quantidade de motoristas e horas extras utilizados, sem contudo violar as restrições operacionais e trabalhistas. À cada seqüência de viagens associada a um motorista damos o nome de *jornada*, que pode designar também uma seqüência de atividades ligadas a um veículo.

A diferença entre os escalonamentos de tripulação no curto[3] e longo[4] prazos é que, no último, devem ser considerados, entre outros aspectos, folga semanal, férias e feriados. No PPV, considera-se apenas o planejamento diário das linhas e, neste caso, apenas o escalonamento de tripulação de curto prazo é relevante.

Em alguns países, um motorista pode utilizar mais de um veículo[5] mas, como no Brasil esta situação normalmente não ocorre, este trabalho a desconsidera. No entanto, em algumas cidades brasileiras, podem existir jornadas de tipo *dupla pegada*, quando as atividades dividem-se em dois turnos. Dependendo dos acordos sindicais, jornadas dupla pegada ficam proibidas ou restritas a uma pequena porcentagem das jornadas. O tratamento do caso de dupla pegada será discutido ao longo do texto.

1.2.4 Restrições do Problema

As restrições do PPV são divididas em duas classes: as relativas à frota e as relativas à mão-de-obra.

Restrições relativas à frota

- O número de viagens partindo de cada um dos PCs em uma faixa horária deve ser tal que atenda à demanda de passageiros naquele sentido da linha e dentro de um certo padrão de qualidade.

- Os horários de partida dos veículos devem estar bem distribuídos ao longo do dia.

- O veículo deve sair da garagem antes da primeira viagem realizada por ele e voltar depois da última viagem para a mesma garagem.

[3] *crew scheduling.*
[4] *crew rostering.*
[5] a troca de veículos durante uma jornada é conhecida por *changeover.*

Restrições relativas à mão-de-obra

- A jornada de trabalho diária normal de um motorista é de 7 horas e 20 minutos.
- Há um limite no número de horas extras que um funcionário pode fazer por dia.
- A quantidade de jornadas do tipo dupla pegada deve estar limitada a uma certa fração do total de motoristas necessários.
- Todo motorista tem o direito a um descanso com duração de pelo menos 30 minutos ao longo do dia, entre a $2^{\underline{a}}$ e a $6^{\underline{a}}$ hora de sua jornada. É possível optar pela não concessão do descanso mas, neste caso, o funcionário deve ter sua jornada encurtada em pelo menos 30 minutos em relação à jornada normal de trabalho.
- Uma *rendição* ocorre quando um motorista inicia sua jornada em um PC tomando o veículo de outro motorista que, por sua vez, termina sua jornada. Nesta situação, os dois motoristas envolvidos devem dedicar 20 minutos a essa atividade.

Normalmente, descanso é considerado o intervalo de tempo entre o término de uma viagem e o início da viagem seguinte. Como todo o trabalho dá enfoque a esse descanso de pelo menos 30 minutos, é a ele que se refere o termo *descanso* a partir de agora.

A representação de todos as restrições do PPV na formulação matemática do problema pode levar a modelos que não são tratáveis computacionalmente. Assim, neste trabalho, procurou-se isolar um conjunto de restrições principais que devem ser atendidas pelas soluções geradas pelos modelos. Deste modo, algumas restrições não foram explicitamente impostas nos modelos aqui propostos. Estas restrições que foram desconsideradas são listadas na subseção a seguir. Em termos práticos, espera-se que estas restrições sejam mais brandas e acabem sendo satisfeitas pelas soluções oriundas dos modelos propostos.

Restrições não consideradas

- Devido à limitação de espaço físico, o número de ônibus simultaneamente estacionados em um PC não pode ultrapassar uma determinada marca. O modelo proposto não garante o atendimento a esta restrição apesar de o algoritmo levá-la implicitamente em consideração em algumas etapas do planejamento. Além disso, em muitos casos, o acúmulo de ônibus nos PCs não constitui um problema muito grave no planejamento operacional das linhas.
- Em algumas linhas, o descanso de um motorista não pode ser feito em determinado PC. O sistema não prevê estes casos e permite que os descansos ocorram em qualquer PC.

- Não há qualquer restrição obrigando que as jornadas tenham um tamanho mínimo. Jornadas com apenas uma viagem entre PCs, por exemplo, podem fazer parte da solução.

- Ainda, pode haver restrições adicionais específicas de uma empresa ou de uma localidade, mas o sistema trata somente as mais gerais.

Os dados referentes, por exemplo, ao limite de horas extras permitidas ou à duração mínima do descanso podem variar conforme a empresa ou a localidade. Assim, ao longo do texto e dependendo da linha, a jornada máxima de trabalho pode ser diferente de 7 horas e 20 minutos e o tamanho mínimo de descanso diferente de 30 minutos. Estes valores são fornecidos através de parâmetros do sistema, como mostra a subseção a seguir.

1.2.5 Dados de Entrada

Os dados de entrada dizem respeito a uma linha de ônibus específica e a um tipo de dia (útil, feriado ou fim de semana). São eles:

1. Uma tabela de demanda que mostra o número de passageiros a serem atendidos por faixa horária do dia e por sentido (de um PC para o outro e vice-versa).

2. O número máximo de passageiros que podem ocupar simultaneamente um ônibus dentro do padrão de qualidade exigido.

3. Os tempos aproximados de duração das viagens entre os PCs e entre um PC e a garagem, por faixa horária e por sentido de viagem.

4. Porcentagem máxima de jornadas que podem ser do tipo dupla pegada.

5. Quantidade máxima de minutos que um motorista pode trabalhar além do período normal de jornada.

6. Duração da rendição.

7. Tamanho normal da jornada de trabalho.

8. Duração mínima de descanso.

9. Intervalo após o início da jornada em que o motorista pode descansar.

1.3 Sobre os Algoritmos

Os algoritmos apresentados no documento usam uma pseudo-notação de programação lógica ou de linguagem procedural, visto que foram desenvolvidos em uma linguagem de programação lógica que possui estruturas típicas de linguagens imperativas, como por exemplo 'se então'.

Os algoritmos, em sua maioria, foram simplificados de tal forma que pudessem ser mais facilmente compreendidos. Pelo mesmo motivo, alguns deles são apresentados de maneira diferente da usada na implementação real, porém permitindo que a idéia central de sua corretude seja mais facilmente entendida.

No capítulo 4, os predicados são freqüentemente referenciados por seu nome e sua aridade. Por exemplo, um predicado com três argumentos chamado *labeling* pode ser identificado por *labeling/3*.

Capítulo 2

Técnicas

Em problemas de otimização combinatória, procura-se a melhor solução sobre um espaço de soluções viáveis. O escalonamento de tripulação e de veículos são problemas de otimização combinatória de grande escala, isto é, onde o espaço de soluções viáveis contém uma quantidade bastante elevada de elementos.

Em [26], a autora destaca a habilidade da programação por restrições [31, 52, 53] para representar modelos flexíveis e resolver pequenos problemas com restrições complexas. Ressalta também a importância de se utilizar abordagens da programação matemática e algoritmos clássicos de pesquisa operacional para resolver eficientemente problemas maiores.

Não raro, quando se utiliza programação matemática para resolver um problema de grande porte, procura-se reduzi-lo a um outro problema de otimização combinatória (por exemplo, partição de conjuntos, cobertura de conjuntos, emparelhamento[1]) que possua um modelo bem estudado [26]. Os algoritmos de programação matemática podem resolver modelos de problemas de otimização com milhares de variáveis e restrições, mas nem sempre são esses os modelos mais adequados. Por outro lado, nem sempre é possível resolver o modelo exatamente usando programação matemática. Nestes casos, as heurísticas [43] são geralmente usadas para auxiliar na busca de soluções viáveis para o problema permitindo que o modelo seja resolvido, ainda que de forma aproximada, em tempo razoável. A seguir, apresentamos conceitos básicos referentes às técnicas empregadas na resolução do PPV e as ferramentas utilizadas neste trabalho.

[1] Esses problemas são conhecidos respectivamente por *set partitioning*, *set covering* e *matching flow*.

2.1 Programação por Restrições

Muitos problemas combinatórios, como alocação de recursos e escalonamento de tarefas, podem ser representados como problemas de programação por restrições (CSPs[2]). Um CSP consiste de um conjunto de variáveis, um domínio associado a cada variável determinando os possíveis valores que ela pode tomar, e um conjunto de restrições limitando os valores das variáveis quando combinadas. Uma solução para um CSP é uma atribuição a cada variável de um valor de seu domínio, de tal forma que todas as restrições sejam satisfeitas. No entanto, muitas vezes isso não é o suficiente, quando o objetivo é encontrar uma solução ótima em relação a uma determinada função objetivo. Neste caso, além das restrições que devem ser satisfeitas, ainda deve haver uma outra que leve à minimização da função objetivo.

A estrutura básica de um modelo de programação por restrições é a seguinte:

<Declaração de Variáveis e Domínios>

<Imposição de Restrições>

<Busca por Soluções>

Normalmente, os domínios definidos para as variáveis de um CSP são maiores do que a quantidade de valores que as variáveis realmente podem receber, de maneira a satisfazer as restrições do problema. Assim, utiliza-se mecanismos de propagação de restrições para eliminar valores dos domínios das variáveis que não fazem parte de nenhuma solução.

Consideremos um exemplo simples do resultado da propagação de restrições, citado em [54]. Suponha que três variáveis, com domínio inteiro [1..100] estão sujeitas à restrição $X + Y = Z$. Após a propagação, os domínios das variáveis são alterados como mostra a tabela 2.1. O valor um não faz mais parte do domínio de Z pois a soma de dois números maiores ou iguais a um não pode ser unitária. Da mesma forma, se uma das parcelas fosse 100, o resultado da soma obrigatoriamente seria um valor que não se encontra no domínio de Z. Assim, os domínios de X e Y não podem envolver o valor 100.

Dados		Propagação
$X \in 1, \ldots, 100$		$X \in 1, \ldots, 99$
$Y \in 1, \ldots, 100$	$\Rightarrow$	$Y \in 1, \ldots, 99$
$Z \in 1, \ldots, 100$		$Z \in 2, \ldots, 100$

Tabela 2.1: Propagação de restrições

Se após a propagação de restrições o domínio de alguma variável resultar vazio, o pro-

[2] *constraint satisfaction problems.*

blema não apresenta solução viável, pois não há combinações de valores para as variáveis que satisfazem todas as restrições. Por outro lado, se o domínio de cada variável converge para um único valor, a única solução possível é a atribuição destes valores às variáveis correspondentes. Caso não ocorra nenhuma destas situações, é necessário que um método de busca seja utilizado para encontrar as soluções viáveis do CSP.

Os métodos de busca podem variar muito quanto à maneira como procuram soluções para os problemas. Cada método tem uma maneira particular de escolher a ordem com que as variáveis são avaliadas e a ordem com que os valores do domínio de uma variável são testados. Eles podem varrer todo o espaço de busca ou não, e ainda podem utilizar mecanismos de propagação de restrições embutidos na busca. O método de busca, então, deve ser escolhido de acordo com as características do problema.

Portanto, para encontrar soluções viáveis para um CSP, a programação por restrições, normalmente, deve usar informação contida na estrutura do problema para reduzir o espaço de busca. Em seguida, usa um método de busca para encontrar as soluções ou garantir a não existência de soluções. Esse processo de busca pode ser guiado pelo usuário com o conhecimento que ele tem sobre a estrutura do problema.

2.2 Programação Linear

A programação linear [3, 56] lida com problemas de minimização ou maximização de uma função linear na presença de restrições representadas por equações e inequações também lineares. Os modelos lineares representam muitos sistemas reais de uma forma bastante satisfatória e são capazes de prover uma grande quantidade de informações. Eles ainda podem ser utilizados para resolver certos tipos de problemas de otimização não lineares através de aproximações lineares sucessivas.

Um programa linear é um problema que pode ser expresso da seguinte maneira:

$$\begin{array}{ll} \textit{Minimize} & cx \\ \textit{Sujeito a} & Ax = b \\ & x \geq 0 \end{array}$$

onde x é o vetor de variáveis a serem instanciadas, A é uma matriz de coeficientes conhecidos e c e b são vetores de coeficientes conhecidos. A expressão cx é denominada função objetivo e as equações $Ax = b$ e $x \geq 0$ são as restrições.

Problemas de *programação linear inteira* são problemas de programação linear nos quais variáveis de decisão estão restritas a receberem valores inteiros. O problema é de *programação linear inteira mista* se nem todas as variáveis assumem valores inteiros.

Diferentemente da programação linear, não há algoritmos polinomiais conhecidos para se resolver um problema genérico de programação linear inteira. É preciso portanto, re-

correr a algoritmos como o *branch-and-bound* [40], que, apesar de possuírem complexidade exponencial, percorrem o espaço de soluções de forma criteriosa.

2.3 Programação por Restrições x Programação Linear Inteira

Os sub-problemas do PPV podem ser representados por sistemas lineares em domínios finitos. Para resolvê-los e otimizá-los, podem ser utilizados algoritmos de programação linear inteira (PLI) e programação por restrições (PR) em domínios finitos. Nenhuma dessas técnicas é a mais adequada em todas as situações, pois as características do problema devem ser levadas em consideração. Mesmo diferentes instâncias de certo problema podem apresentar comportamentos muito diferentes, o que torna complicado escolher qual das técnicas utilizar em uma determinada aplicação.

Existem algumas vantagens em representar um problema como um CSP ao invés de representá-lo como um problema de PLI. Um CSP pode representar melhor um problema, já que as restrições não precisam ser expressas em desigualdades lineares. Conseqüentemente, torna-se mais fácil modelar o problema. Além disso, apesar dos modelos de CSPs serem essencialmente mais simples, às vezes eles podem encontrar soluções viáveis mais rapidamente que os métodos de PLI [48].

Também é fácil eliminar simetria em modelos CSPs durante a busca ou através de restrições, o que é uma tarefa complicada para a PLI, pois as restrições que eliminam a simetria podem aumentar bastante o tamanho do modelo. Isto ilustra uma das principais diferenças entre os dois métodos: enquanto os algoritmos de PLI cortam o espaço de busca globalmente, a PR o reduz localmente.

Os algoritmos de PLI fazem poda do espaço de busca baseando-se na geração de limitantes duais e primais e, tipicamente, funcionam muito bem quando o modelo linear relaxado é capaz de produzir limitantes duais fortes. Nesses casos, a PLI pode até levar à solução exata do problema. Já a PR, em geral, não tem esta característica de gerar limitantes duais e, por isso, tem dificuldades em provar otimalidade de soluções. Por outro lado, o mecanismo de busca acelera a obtenção de soluções viáveis, que é o ponto mais fraco dos algoritmos de PLI.

A literatura especializada sugere que o uso da PR é mais adequado nos seguintes casos:

- Tratamento de simetria [4, 41, 25].
- Modelos com restrições não-lineares [41].
- Problemas para os quais a formulação básica é trivial em programação por restrições,

mas que requerem o uso de variáveis e restrições não naturais ao problema para serem modelados em programação inteira [4].

- O problema é bem-conhecido e difícil mesmo para exemplos pequenos [4].
- Situações em que uma solução viável é tudo o que se precisa, sem a necessidade de ser ótima[50].
- Problemas com muitas restrições, para os quais provavelmente não há soluções[3] [2].
- Modelos com restrições de aridade[4] baixa [41].
- Busca por soluções em espaço de busca pequeno [2].

Por outro lado, modelos PLI conseguem resolver, eficientemente e de forma exata, problemas de otimização bastante grandes, o que, normalmente, não pode ser feito usando PR. CSPs freqüentemente são resolvidos com métodos de busca local [1], que basicamente consistem em criar uma solução aleatória ou heurística para um problema e modificá-la pouco a pouco, até torná-la uma solução boa e sem usar muito tempo de computação. No entanto, desenvolver um método de busca local eficiente pode ser uma tarefa trabalhosa e demorada.

A comparação entre PR e PLI ainda precisa de maiores investigações [41, 50, 11], mas se for possível aproveitar as vantagens de cada uma, a integração entre as duas [4] muitas vezes torna-se a melhor opção para se resolver um problema de otimização combinatória.

2.4 Ferramentas Utilizadas

O resolvedor de programação por restrições utilizado implementa restrições em domínios finitos. É o que atende às necessidades do PPV da melhor forma, apesar de haver outros resolvedores que consideram, por exemplo, restrições em conjuntos, em aritmética linear e restrições booleanas [14].

Dentre as linguagens que suportam programação por restrições, o ECLiPSe [34, 55] foi escolhido porque, além de outros fatores apontados em [13], é um *software* de baixo custo e baseado em programação lógica [33]. As vantagens de se utilizar programação por restrições suportada por linguagens de programação lógica são expostas em [24, 54, 55].

A biblioteca FD (*finite-domain*) [14] do ECLiPSe pode ser vista, segundo [55], como a junção de três bibliotecas. A primeira manipula variáveis com domínios finitos simbólicos, como {*vermelho, amarelo, azul*}, e restrições de igualdade e desigualdade entre expressões

[3] problemas *overconstrained*.

[4] A aridade de uma restrição corresponde ao número de variáveis afetadas por ela.

que são constantes ou variáveis. Estas restrições também podem ser usadas quando os domínios são numéricos. A segunda biblioteca manipula restrições numéricas envolvendo variáveis inteiras e relações de igualdade e desigualdade entre expressões lineares. E, por fim, a terceira biblioteca suporta restrições complexas pré-definidas, como por exemplo *alldifferent*, que obriga um conjunto de variáveis a assumir valores diferentes.

Vários problemas de programação linear também compõem o sistema implementado neste trabalho. Para resolvê-los, utiliza-se o ILOG CPLEX 7.0 [29], que é uma das melhores ferramentas computacionais para se resolver problemas de programação linear, especialmente quando combinado à linguagem de modelagem algébrica AMPL [21, 28].

Uma *linguagem de modelagem* expressa o modelo fazendo uso de uma notação mais abstrata e um compilador traduz essa notação para a forma de entrada do resolvedor. Uma *linguagem de modelagem algébrica* baseia-se na notação matemática tradicional para descrever objetivos e restrições. As principais vantagens de uma linguagem de modelagem algébrica são:

- Torna a programação matemática mais econômica e confiável.
- Pode ser aplicada a uma grande variedade de modelos de programação linear, inteira e não-linear.
- Sua notação é familiar, pois é facilmente compreendida por quem já conheça álgebra ou cálculo.

A análise das soluções provou ser muito importante durante o desenvolvimento do sistema. Essa análise foi facilitada e agilizada com o uso de uma ferramenta de visualização desenvolvida no Instituto de Computação da Universidade Estadual de Campinas (UNICAMP) [51]. Ela exibe de forma clara e intuitiva a tabela de horários e as escalas de ônibus e de motoristas, além de informações detalhadas referentes a cada ônibus e a cada motorista. O *software* ainda permite a alteração dos dados de entrada e a manipulação das soluções de maneira simples.

Em resumo, as principais ferramentas utilizadas neste trabalho são as seguintes:

- A biblioteca FD na linguagem ECLiPSe.
- O resolvedor de programação linear CPLEX.
- A linguagem de modelagem algébrica AMPL.
- O *software* de visualização desenvolvido na UNICAMP.

Capítulo 3

Abordagens para resolver o PPV

A maioria dos algoritmos propostos para resolver o PPV utilizam uma abordagem seqüencial (veja figura 3.1) em que, primeiramente, a tabela de horários é montada e depois são efetuados separadamente os escalonamentos de ônibus e de motoristas, seguindo esta ordem. No caso dos problemas que também envolvem o *crew rostering*, este é resolvido numa última etapa. Em [22, 23], os autores não só propõem uma abordagem integrada dos escalonamentos de veículos e de tripulação, como a comparam com o algoritmo seqüencial e destacam os benefícios desta integração.

A montagem dos horários de partida das viagens é o primeiro passo de todos os algoritmos propostos. Normalmente, os horários de partida são calculados a partir da análise da demanda ao serviço e não são mais alterados durante a execução do algoritmo.

Os escalonamentos de veículos e de tripulação são claramente relacionados já que as decisões tomadas em um destes problemas afetam o outro de maneira considerável. Assim, quando a resolução é feita de maneira seqüencial, o segundo problema a ser resolvido deve atender também às restrições impostas pela solução do primeiro, o que pode impedir a descoberta de soluções mais interessantes.

As restrições impostas ao escalonamento de tripulação são consideravelmente mais rígidas do que as impostas ao escalonamento de veículos, o que torna este último um problema bem mais flexível. Assim, se este é efetuado antes, ainda mais restrições devem ser atendidas no escalonamento de tripulação. Se os problemas são resolvidos na ordem oposta, novas restrições são adicionadas ao escalonamento de veículos. Porém, alguns fatores influenciam para que o escalonamento de veículos não seja muito afetado pela solução do escalonamento de tripulação. No caso brasileiro em particular, isto se deve principalmente ao fato de que cada linha é tratada de forma isolada e cada motorista só opera um ônibus. Com isso, como são poucas as restrições específicas sobre os veículos, se o problema de escalonamento de tripulação for resolvido de modo a minimizar o número de motoristas, é razoável supor que a quantidade de carros necessária a ser disponibilizada

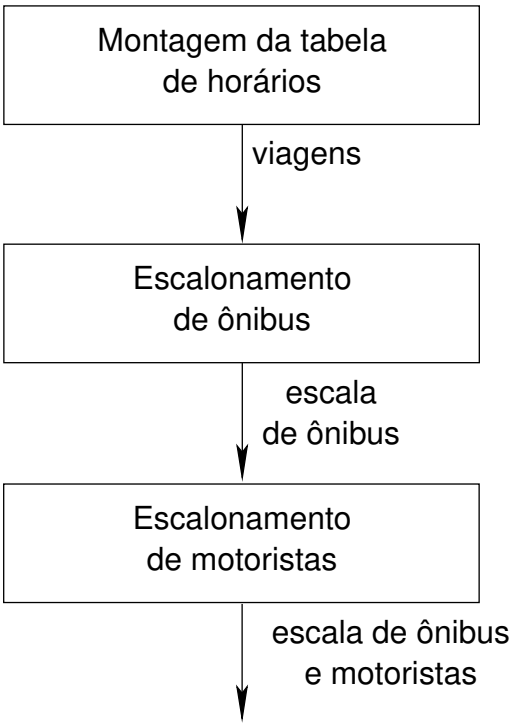


Figura 3.1: Abordagem seqüencial

aos motoristas também se encontre próxima da mínima. Ou seja, otimizar-se-á também o problema de escalonamento de veículos.

Portanto, quando se adota uma abordagem seqüencial em um cenário como o brasileiro, uma boa opção é fazer o escalonamento de tripulação antes do de veículos. Contudo, normalmente, faz-se o contrário resolvendo-se o problema mais fácil primeiro. A solução deste restringe o problema mais difícil, de escalonamento de tripulação, tornando-o mais fácil, porém descartando soluções que possivelmente sejam melhores.

Assim, ao que tudo indica, a melhor maneira de se resolver o PPV seria com a integração total dos dois problemas. No entanto, esta não é uma tarefa fácil e, dependendo do modelo, a integração pode não ser compensadora, pois pode aumentar em muito a complexidade do problema.

Em [22, 23], os algoritmos são classificados em diferentes categorias dependendo da estratégia que adotam para solucionar estes dois problemas de escalonamento. Estas estratégias incluem:

1. Escalonamento de veículos durante uma abordagem heurística para o escalonamento de tripulação.

2. Inclusão de considerações da tripulação no processo de escalonamento de veículos;

o escalonamento de tripulação só é realmente feito depois.

3. Integração completa dos escalonamentos de veículos e de tripulação.

O algoritmo proposto neste trabalho não adota nenhuma destas estratégias, pois faz primeiramente o escalonamento de tripulação e depois o escalonamento de veículos.

Usualmente, o PPV é tratado como um processo com três etapas distintas: montagem da tabela de horários, escalonamento de motoristas e escalonamento de veículos. A seguir, faz-se uma breve revisão dos algoritmos propostos anteriormente na literatura para resolver cada uma destas etapas.

3.1 Montagem da Tabela de Horários

A construção da tabela de horários de uma linha de ônibus baseia-se na análise da demanda ao serviço, mas pode ser feita de formas diferentes, de acordo com os dados disponíveis. Em [35, 39], por exemplo, os horários de partida das viagens são calculados a partir da informação de cada passageiro levando-se em consideração os instantes em que ele gostaria de sair e em que ele realmente saiu. A distribuição ótima de viagens minimiza a soma dos custos dos passageiros, que são proporcionais aos seus tempos de espera. Analisa-se o cenário em que são distribuídos uniformemente os horários de viagens em que os passageiros desejam partir.

A tabela de horários também pode ser construída seguindo-se uma distribuição ideal de viagens. Além de satisfazer as necessidades de demanda do problema, essa distribuição de viagens é muito simples de calcular e exige informações fáceis de se obter. Em uma *distribuição ideal de viagens* todos os tempos de espera entre as viagens de uma mesma faixa horária são iguais e a metade deste tempo de espera corresponde ao espaçamento entre as viagens das extremidades e as fronteiras da faixa horária. Entenda-se por *viagens das extremidades* a primeira e a última viagens de determinada faixa horária em um PC. A distribuição ideal ainda deve atender à demanda dentro do padrão de qualidade exigido. A figura 3.2 mostra um exemplo de distribuição ideal em uma linha que tem 6, 5 e 3 viagens partindo em 3 faixas horárias consecutivas de determinado PC.

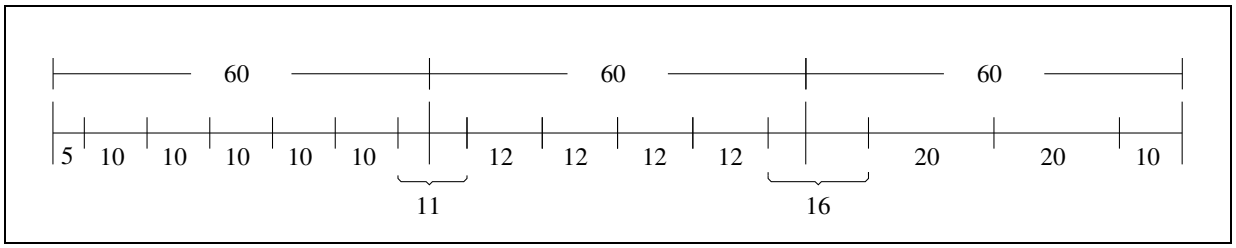


Figura 3.2: Distribuição ideal

Portanto, os horários de partida de viagens em um PC na distribuição ideal são estabelecidos pela seguinte fórmula:

$$V_{i,f} = 60/(2 * N) + 60/N * (i - 1) + I_f,$$

em que $V_{i,f}$ é a i-ésima viagem na faixa horária f, N é a quantidade de partidas em f, e I_f é o minuto inicial de f.

A *distribuição ideal mínima de viagens* é a distribuição ideal que utiliza o menor número de viagens necessário para atender à demanda. A quantidade mínima de viagens para cada intervalo de tempo é facilmente calculada a partir da demanda e da capacidade dos ônibus.

Distintamente dos outros trabalhos, os horários das viagens em [46] são determinados simultaneamente com o escalonamento de motoristas e de ônibus. Uma tabela de horários é criada inicialmente, mas com a diferença de que as viagens não precisam obrigatoriamente partir nestes horários, denominados *horários de aderência*. Porém, o modelo premia cada viagem que parte em um horário de aderência e penaliza cada viagem que não coincide com um horário de aderência. Apesar de ter conseguido bons resultados, a distribuição dos horários de partida das viagens não é totalmente satisfatória. Para resolver este problema, [46] criou uma heurística gulosa simples que procura realizar pequenos deslocamentos das viagens para antes ou depois do seu horário de partida, com o intuito de espaçar melhor os tempos de partida das viagens em cada um dos PCs. Os resultados deste trabalho motivaram o desenvolvimento de um algoritmo que retorna a distribuição ótima de horários das viagens a partir das escalas de ônibus e de motoristas (veja seção 4.4). As novas soluções obtidas mostram que a distribuição de viagens pode melhorar muito quando se usa o novo algoritmo. Portanto, é possível resolver o PPV a partir de uma tabela inicial de horários não definitiva, mas a qualidade da distribuição de horários na solução final depende da flexibilidade do sistema quanto à alteração da tabela inicial após e durante o escalonamento de motoristas e de ônibus.

3.2 Escalonamento de Motoristas

Em [59], é feito um levantamento de alguns dos principais sistemas de escalonamento de motoristas. Na época dos primeiros sistemas, tais como RUCUS, HOT e TRACS, não havia poder computacional suficiente para se utilizar programação matemática plenamente. Então, eles utilizavam métodos heurísticos para gerar uma solução inicial de jornadas e para melhorá-la na tentativa de encontrar uma boa solução final.

Com o avanço do poder computacional, modelos de programação matemática passaram a ser utilizados para resolver mais eficientemente os problemas de escalonamento de

tripulação. No entanto, métodos heurísticos continuam sendo utilizados [57] para, por exemplo, reduzir o tamanho dos problemas, que geralmente são enormes. Os sistemas de escalonamento de motoristas que obtiveram maior sucesso, como TRACSII, HASTUS e EXPRESS, são baseados em programação matemática combinada com heurísticas.

Dentre as três abordagens apresentadas no começo do capítulo, a mais utilizada é a primeira, em que o escalonamento de veículos é feito durante uma abordagem heurística para o escalonamento de tripulação. Um conjunto enorme de jornadas é gerado e um subconjunto deste, suficiente para realizar todas as viagens, é selecionado de acordo com uma função objetivo, utilizando-se um modelo de partição ou cobertura de conjuntos. No entanto, se todas as jornadas válidas fossem consideradas, o problema ficaria muito grande e seria impossível resolvê-lo em um intervalo de tempo razoável, o que torna necessário o uso de heurísticas para gerar e selecionar boas jornadas iniciais.

A diferença entre utilizar partição e cobertura de conjuntos é que, no primeiro caso, não se permite que uma viagem seja coberta por mais de um motorista, ou seja, a tripulação nunca faz papel de passageiro. Por ser mais fácil encontrar solução para a cobertura de conjuntos, esta é muitas vezes a escolhida, até porque é possível diminuir o número de motoristas que atuam como passageiros[1] ao final do processo. Portanto, o escalonamento de motoristas é normalmente resolvido da forma descrita no início do capítulo 2, pois trata-se de um problema grande de otimização que geralmente é resolvido com programação matemática utilizando modelos bem definidos [26], como é o caso da partição e da cobertura de conjuntos.

Escalonamento de motoristas no Brasil

Alguns dos principais sistemas de escalonamento de motoristas já foram testados em cidades brasileiras. A seguir, explica-se o funcionamento de dois destes sistemas e apresenta-se os resultados gerados por eles.

O IMPACS faz parte do módulo de escalonamento de tripulação do sistema comercial BUSMAN e é capaz de resolver problemas grandes utilizando uma formulação de cobertura de conjuntos com restrições limitando a quantidade de jornadas de certos tipos. O tamanho do problema é reduzido através de uma série de rotinas que geram as jornadas e eliminam as menos eficientes. Ao final, trocas heurísticas de atividades entre jornadas são realizadas a fim de se obter uma melhora na solução.

O IMPACS adaptado ao cenário brasileiro foi testado na cidade de Fortaleza [15] e alcançou uma redução de custos de até 12% em relação à solução manual, com escalas que exigiram 16 ônibus e 31 motoristas.

O TRACSII [20, 18] é um sistema muito utilizado no escalonamento de motoristas

[1] Dá-se o nome de *over-cover* à atribuição de uma mesma viagem a mais de um motorista.

de empresas de ônibus e trens, em que se identifica basicamente três estágios. Nos dois primeiros são utilizados métodos heurísticos para, primeiramente, gerar um conjunto de jornadas válidas e, então, reduzi-lo a um subconjunto deste para que o problema torne-se tratável. Finalmente, seleciona-se, através da programação linear inteira, uma escala destas jornadas que cubra as atividades dos veículos. O modelo utilizado neste último passo é o de cobertura de conjuntos. Em São Paulo, uma instância foi testada no TRACSII [58] utilizando-se diferentes valores para alguns parâmetros. Na média, cada experimento levou em torno de cinco minutos em um Pentium de 110MHz, e as soluções encontradas foram muito parecidas com as que eram praticadas. Em Sorocaba, o mesmo sistema foi utilizado e atingiu um ganho de mais de 6% no número de motoristas e de 2% em horas extras.

Programação por restrições no escalonamento de tripulação

A combinação de heurísticas e programação matemática tem sido eficiente na resolução do escalonamento de tripulação, mas ainda apresenta falhas, o que estimula a busca por novos métodos. Nessa busca, a programação por restrições passou a ser considerada há alguns anos.

Em [8], é feito um levantamento de vários estudos relativos à utilização de técnicas de programação por restrições para resolver problemas de escalonamento de tripulação. Percebe-se que muitas pesquisas nessa área tratam escalas de tripulação em companhias aéreas. Ademais, elas dependem muito do uso de soluções de programação linear para guiar a ordenação de variável e valor. Em [27] por exemplo, resolve-se o programa linear associado ao sistema de partição de conjuntos, o que resulta em uma solução ótima fracionária, com valores entre 0 e 1. As variáveis do modelo de programação por restrições então, são ordenadas de acordo com estes valores, sendo a primeira a que tem o valor mais próximo de 1 e assim por diante. A cada variável é atribuído o valor 1, e depois 0, caso ocorra *backtracking*. Em [44], as variáveis que estão à menor distância de 0 ou 1 são rotuladas antes, e o valor inteiro que está mais próximo - 0 ou 1 - é o primeiro a ser escolhido na instanciação.

Em [27], um modelo de partição de conjuntos foi utilizado para resolver o problema de escalonamento de tripulação para empresas aéreas. O programa utilizou programação por restrições. No entanto, as ordenações de variáveis e valores foram guiadas pela solução do modelo de programação linear relaxado. O sistema não alcançou resultados equiparáveis aos sistemas que utilizam programação matemática pura.

Em [44], foi concebida uma maneira geral de se combinar programação matemática e programação por restrições com o intuito de resolver eficientemente problemas difíceis que levam muito tempo para serem resolvidos utilizando isoladamente programação por restrições ou programação matemática. Ela resolve o problema linear relaxado e usa essa

solução para ordenar as variáveis. O sistema foi testado em um problema de escalonamento de tripulação aérea e apresentou desempenho pior do que uma abordagem utilizando exclusivamente programação matemática, mas bem melhor do que uma abordagem utilizando somente programação por restrições.

Os dois sistemas citados acima e ainda o construído por [36] utilizaram modelos de partição de conjuntos e propagação de restrições para produzir escalas de tripulação aérea. Todos estes sistemas funcionaram. Porém, os problemas tratáveis por eles são bem menores do que os problemas que podem ser resolvidos por sistemas baseados em programação matemática. No entanto, nota-se que as vantagens da programação por restrições não foram muito exploradas nestes casos, já que os modelos utilizados favorecem o uso de programação matemática.

O escalonamento de motoristas foi formulado, em [11], como um problema de cobertura de conjuntos, em que as variáveis são porções de trabalho[2] e o domínio de cada porção é o conjunto de índices das jornadas que o cobrem. As *porções de trabalho* são seqüências de atividades que, de acordo com o escalonamento de veículos, devem ser feitas por um mesmo veículo e as *atividades* correspondem às viagens necessárias para atender à demanda e às viagens que partem da garagem ou terminam nela. A utilização de programação por restrições neste modelo não gerou resultados satisfatórios.

Em [61], uma abordagem híbrida de programação por restrições e programação inteira foi utilizada para resolver o escalonamento de motoristas. A programação matemática é utilizada para resolver uma cobertura de conjuntos enquanto a programação por restrições gera as jornadas para o problema. Em vez de utilizar jornadas fixas como nos demais estudos, o sistema utilizou regras para construir jornadas com programação por restrições e alcançou bons resultados.

Em [9], a intenção foi testar a programação por restrições no problema de escalonamento de motoristas. O modelo de partição de conjuntos foi utilizado no lugar da cobertura de conjuntos pois gera maior propagação de restrições e explora melhor a estrutura do problema. O modelo testado considerou as porções de trabalho como variáveis, assim como em [11], e também usou programação linear para guiar a busca no modelo de programação por restrições. Este sistema obteve relativo sucesso. No entanto, sistemas baseados em programação linear inteira, como o TRACSII, são mais rápidos e podem resolver problemas maiores.

Nota-se que as tentativas de substituir a programação inteira por programação por restrições em modelos similares não parecem produzir melhoras nos resultados, nem mesmo quando se utiliza a solução da programação linear para guiar a busca no modelo de programação por restrições. Já o algoritmo proposto por [61] explorou vantagens da programação por restrições para melhorar as soluções geradas por um sistema baseado em

[2] referenciados na literatura por *pieces of work*.

programação matemática. Da mesma forma, o presente trabalho pretende unir o que há de melhor em ambas as abordagens.

Técnicas usadas no escalonamento de tripulação

Muitas técnicas já foram testadas em problemas de escalonamento de motoristas. Geração de colunas [12, 22, 19, 61] e relaxação lagrangeana, por exemplo, são técnicas de programação matemática que têm obtido sucesso. Também têm sido experimentadas muitas maneiras de se aplicar métodos heurísticos, como algoritmos genéticos [6] e redes neurais [8]. Este trabalho objetiva conceber modelos diferentes dos já existentes utilizando as funcionalidades básicas da programação por restrições e da programação matemática.

3.3 Escalonamento de Ônibus

Normalmente, o escalonamento de ônibus é resolvido considerando-se uma única garagem e um único tipo de veículo, o que pode ser feito usando-se um algoritmo polinomial de fluxo em rede.

O escalonamento de ônibus baseado em rede consiste em ligar viagens da melhor forma possível para formar uma seqüência válida de atividades para cada ônibus, e utilizando-se o mínimo de veículos. Também há estudos que consideram um tamanho fixo da frota [38]. No entanto, mesmo nestes casos, é possível encontrar a quantidade mínima de carros necessária para atender à demanda do serviço testando diferentes dimensionamentos de frota. Nestes algoritmos, as viagens são divididas em grupos de maneira que todas as viagens em um mesmo grupo possam ser conectadas para formar uma escala válida de um ônibus, cujo começo e término se dão na mesma garagem. Em [10], mostra-se que o problema já foi formulado de diversas maneiras: como modelos de atribuição, transporte, fluxo em rede, quase-atribuição [37] e emparelhamento [3].

Os modelos propostos anteriormente consideram somente a tabela de horários e alguns parâmetros como entrada. A abordagem utilizada neste trabalho, como poderá ser observado no capítulo 4, efetua o escalonamento de motoristas antes de criar a escala de ônibus, fazendo com que o escalonamento de ônibus seja obrigado a considerar também a escala de motoristas como entrada e não simplesmente a tabela de horários.

Escalonamento de ônibus no Brasil

O transporte coletivo brasileiro apresenta características peculiares que influem no seu planejamento operacional. É comum no Brasil, por exemplo, o uso de apenas um veículo

[3] Estes modelos são conhecidos respectivamente por *assignment, transportation, network flow, quasi-assignment* e *matching*.

por motorista e a cobertura de uma única linha por veículo. Por isso, a estratégia normalmente adotada é, primeiramente, construir escalas de ônibus com intervalos que permitem o motorista fazer sua refeição e continuar conduzindo um mesmo veículo ao longo de toda sua jornada. Em seguida, estes ônibus são utilizados para cobrir as jornadas dos motoristas. Em outros países, estes intervalos não são previstos porque o uso de um mesmo ônibus para linhas diferentes e a condução de ônibus diferentes por um mesmo motorista são aceitos.

Em [58, 15], constrói-se uma escala de ônibus a partir da tabela de horários. A seguir, o sistema procura diminuir o tamanho da frota, fazendo pequenas alterações nas viagens e permitindo que os veículos cubram mais de uma linha. As escalas de ônibus geradas até então pelo sistema utilizado ainda diferem das normalmente produzidas no Brasil, pois não prevêem intervalos para refeição. Na seqüência, calcula-se a escala de motoristas e reformula-se a escala de ônibus para que os motoristas conduzam apenas um veículo ao longo de sua jornada, como ocorre no Brasil, e as viagens que envolvem a garagem sejam estabelecidas corretamente.

Em [58], foram utilizados os sistemas de escalonamento de ônibus BUSPLAN e ALOCA em conjunto com os sistemas de escalonamento de motoristas CREWPLAN e IMPACS, para resolver o problema em Fortaleza. Em São Paulo e Sorocaba, o sistema BOOST [30] fez o escalonamento de ônibus e TRACSII o de motoristas. Em apenas um caso houve ganho no tamanho da frota, de 3.6%. Os resultados referentes à mão-de-obra foram vistos na seção 3.2. ALOCA e BUSPLAN também foram testados em [15]. Estes resultados não surpreendem pelo fato de que, no Brasil, o problema de escalonamento de ônibus é muito restrito devido às suas características e pouco pode ser feito para melhorar uma escala bem construída, mesmo que esta tenha sido gerada manualmente.

ALOCA é um pacote que faz o escalonamento de ônibus procurando simular o método manual, mas pode alcançar resultados mais rápidos e melhores. No entanto, às vezes, gera soluções não operacionais, principalmente quando há um grande número de ônibus circulando somente nos horários de pico. Como foi criado para resolver o problema no Brasil, o sistema prevê intervalos para refeição.

Para gerar uma escala de ônibus, BUSPLAN utiliza o algoritmo VAMPIRES, que procura minimizar o tamanho da frota. O algoritmo gera uma solução inicial não necessariamente válida e depois usa uma heurística para remover as violações às restrições e melhorar a solução.

BOOST (*Basis for Object Oriented Scheduling of Transport*) é um sistema de escalonamento de motoristas que utiliza o algoritmo VAMPIRES modificado para explorar orientação a objeto.

Capítulo 4

O Algoritmo Proposto

O algoritmo proposto neste trabalho foge à maneira usual como o PPV vem sendo tratado em outros trabalhos. Para que a abordagem proposta seja melhor entendida, os principais objetivos do PPV são destacados a seguir:

- Atender à demanda dentro de um certo padrão de qualidade, satisfazendo às restrições operacionais e trabalhistas.
- Utilizar a menor quantidade possível de motoristas, ônibus e horas extras.
- Distribuir o mais uniformemente possível os horários de saída das viagens.

Retomando as observações feitas no início do capítulo 3, o sistema implementado utiliza uma abordagem seqüencial. No entanto, diferentemente da maioria dos sistemas, o escalonamento de motoristas é feito antes do escalonamento de ônibus.

Essencialmente, o algoritmo funciona como mostra a figura 4.1. No primeiro passo, monta uma tabela inicial de horários para depois fazer os escalonamentos de motoristas e de ônibus, nesta ordem, procurando minimizar a quantidade de motoristas, de veículos e de horas extras. Ainda há uma etapa final em que os horários de viagens são redistribuídos, uma vez que podem ser acrescentadas viagens durante os processos de escalonamento de motoristas e de ônibus, quebrando a regularidade da distribuição inicial dos horários.

Cada passo do algoritmo é explicado detalhadamente a seguir.

4.1 Montagem da Tabela Inicial de Horários

Em [46], a tabela de horários é construída simultaneamente com as escalas de motoristas e de ônibus, a partir de uma tabela inicial que pode ser alterada. Isso motivou a implementação de um algoritmo que redistribui os horários de viagens ao final do processo de

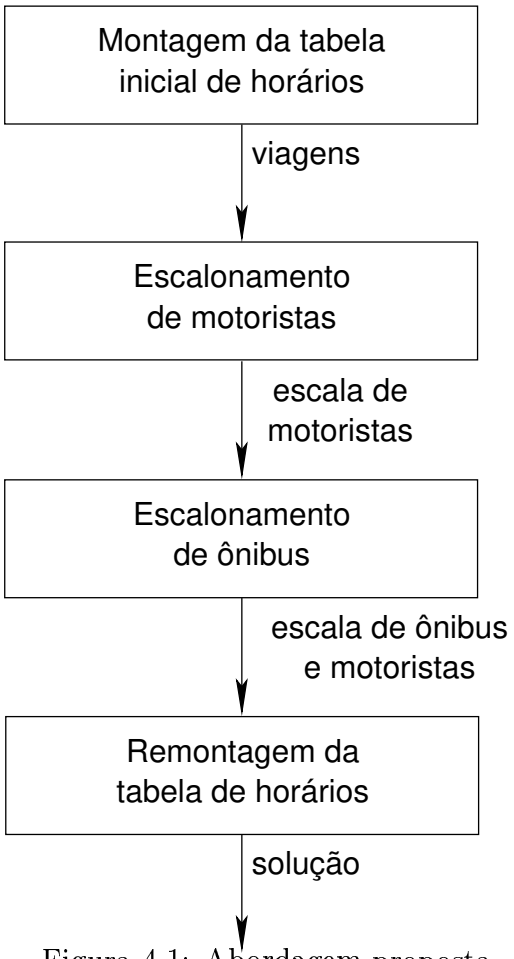


Figura 4.1: Abordagem proposta

resolução do PPV, uma vez que a tabela inicial de horários não precisa ser definitiva. Isso proporciona maior flexibilidade à modelagem do problema, o que poderá ser notado na seção 4.2.1.

Nesta fase, gera-se uma tabela de horários necessária ao modelo de escalonamento de motoristas a partir de um arquivo de entrada que deve conter as seguintes informações:

- Os intervalos de tempo para os quais foram levantados os dados.
- A demanda em cada PC para cada um dos intervalos de tempo.
- As durações das viagens entre PCs e entre PCs e a garagem, em cada intervalo de tempo.
- A capacidade máxima dos ônibus, dentro do padrão de qualidade exigido.

Cabe ao usuário escolher os intervalos de tempo de acordo com os dados disponíveis para as linhas. Os intervalos de tempo podem ser divididos de qualquer forma, desde que a interseção entre eles seja vazia e que todos os minutos entre o início do primeiro intervalo e o fim do último estejam cobertos por algum dos intervalos.

É melhor que um intervalo de tempo seja suficientemente grande para que a demanda correspondente não fique muito abaixo da capacidade máxima dos ônibus, evitando, assim, viagens que transportem poucos passageiros.

Fornecidos os dados de entrada, a tabela de horários é montada seguindo a distribuição ideal mínima, já descrita na seção 3.1. Por construção, essa tabela inicial de horários atende à demanda. O sistema foi desenvolvido de tal forma que viagens nunca são retiradas, mas podem ser acrescentadas ou ter seu horário de partida modificado dentro de seu intervalo de tempo. Assim, se a tabela inicial atende à demanda, qualquer tabela dela derivada também atenderá. Caso nenhuma viagem seja acrescentada ou retirada durante a resolução do PPV, a redistribuição de horários não precisa ser feita, pois a distribuição ideal da tabela inicial é mantida.

4.2 Escalonamento de Motoristas

O escalonamento de motoristas recebe como entrada uma tabela de horários de viagens e os respectivos tempos de duração e deve gerar uma escala de motoristas que atendam a essas viagens. O objetivo é utilizar a menor quantidade possível de motoristas e, com menor prioridade, de horas extras.

A idéia do algoritmo 4.1 é começar procurando uma escala com um motorista. Caso não seja possível, considera-se dois. Novamente, se não for possível, passa-se para três, e assim por diante. Desta forma, a primeira solução encontrada tem o menor número de motoristas que uma solução gerada pelo programa pode alcançar. Como várias soluções podem utilizar o mesmo número de motoristas, procura-se a de menor quantidade de horas extras dentre as soluções que utilizam o mínimo de motoristas.

O problema de escalonamento de motoristas pode ser naturalmente representado por um modelo matricial em que cada variável de decisão $X_{i,j}$ binária representa a relação entre a viagem i e o motorista j. Se $X_{i,j} = 1$, o motorista j é o responsável pela viagem i. Modelos matriciais, assim como variáveis binárias, são muito usados tanto em programação matemática quanto em programação por restrições [17, 42] pois, geralmente, além de fornecerem uma visualização simplificada do problema, permitem fácil identificação e eficiente tratamento de simetrias [16], que são comuns em problemas representados por matrizes.

Três tipos de jornadas de motoristas podem ser encontradas nas empresas:

```
crew_scheduling:-
    dominio(Vars),
    restricoes(Vars),
    busca(Vars).
// estrutura básica de um modelo de programação
// por restrições apresentada na seção 2.1.

busca(Vars):- // A busca começa utilizando um motorista
    busca(Vars,1).
busca(Vars,QtdeMot):-
    instancia(Vars,QtdeMot), !.
busca(Vars,QtdeMot):-
    QtdeMotMais1 is QtdeMot+1,
    busca(Vars,QtdeMotMais1).

instancia(Vars,QtdeMot):-
    retract_all(melhor_solucao(_,_)), // exclui fatos melhor_solucao/2.
    labeling(Vars,QtdeMot), // faz instanciação das variáveis
    custo(Vars,Custo), // calcula o custo de uma instanciação
    (melhor_solucao(_,MelhorCusto), Custo ≤ MelhorCusto → // '→'=então
        true; // ';'=senão
        retract_all(melhor_solucao(_,_)),
        assert(melhor_solucao(Vars,Custo)) // inclui fato melhor_solução/2.
    ),
    fail. // Esta falha faz com que todas as instanciações sejam testadas
instancia(Vars,_):-
    melhor_solucao(Solucao,_), // se não achou nenhuma solução, falha.
    Vars=Solucao. // retorna melhor solução encontrada.
```

Algoritmo 4.1: Escalonamento de Motoristas

Padrão

- A jornada de trabalho normal é de 7 horas e 20 minutos.

- Todo empregado tem direito a um descanso de pelo menos 30 minutos corridos ao longo do dia, todo contido entre a $2^{\underline{a}}$ e a $6^{\underline{a}}$ hora de sua jornada.

- Há um limite pré-estabelecido no número de horas extras que um funcionário pode fazer por dia.

Contínua

- A jornada de trabalho normal é de 6 horas e 50 minutos, o mesmo intervalo de tempo da jornada padrão, subtraído os 30 minutos de descanso.
- O motorista não tem direito ao intervalo de 30 minutos ininterruptos de descanso.
- O motorista não entra em hora extra.

Dupla pegada

- A jornada de trabalho normal é de 7 horas e 20 minutos.
- A jornada é dividida em dois turnos, com intervalo de pelo menos 2 horas entre eles.
- Há um limite no número de horas extras que um funcionário pode fazer por dia.

As jornadas de dupla pegada não foram mencionadas na seção 1.2.4, em que se descreve as restrições do problema, pois normalmente não podem ser utilizadas. Porém, quando permitidas, há um limite pré-fixado para a quantidade de jornadas deste tipo.

Considera-se, na construção da escala de motoristas, que todas as jornadas são padrões. No entanto, elas podem ser transformadas em jornadas dos tipos dupla pegada ou contínua após concluído o escalonamento de motoristas (veja seção 4.3).

4.2.1 Modelo

Todas as restrições descritas na seção 1.2.4 podem ser descritas usando-se somente expressões lineares. As restrições são explicadas detalhadamente após a apresentação dos conjuntos de dados, parâmetros e variáveis.

Conjuntos de dados

Os conjuntos de dados *PC1* e *PC2* definem, respectivamente, as viagens que saem do PC1 e do PC2. O conjunto PCs é formado pelas viagens pertencentes à união de *PC1* e *PC2*.

Parâmetros

n - Define a quantidade máxima de motoristas que podem ser utilizados na escala.
v - A cada viagem $i \in PCs$ é associado um horário de saída v_i.
d - Cada viagem $i \in PCs$ tem um tempo de duração d_i.

d2 - Em certas restrições é necessário saber o intervalo de tempo mínimo que um motorista leva para sair de um PC e voltar para o mesmo. Considerando um motorista que sai no instante de saída da viagem i, este intervalo de tempo é obtido somando-se as durações de ida, d_i e de volta, $d2_i$. Portanto, $d2_i$ é o tempo que levaria uma viagem partindo do PC oposto imediatamente após a chegada da viagem i neste PC.

dgar - Para cada viagem $i \in PCs$, sabe-se o tempo mínimo $dgar_i$ que um ônibus levaria para ir da garagem ao PC de onde parte i.

dgar2 - Para cada viagem $i \in PCs$, sabe-se o tempo mínimo $dgar2_i$ que um ônibus levaria para ir do PC onde termina i à garagem.

fh - Cada viagem $i \in PCs$ começa em um instante que está dentro de uma certa faixa horária fh_i.

p - Para cada motorista j, indexado entre 1 e n, $p_j = 1$ se j está disponível para ser usado na escala, e $p_j = 0$ se não está. Assim, para m motoristas disponíveis, $p_1 = p_2 = \ldots = p_m = 1$ e $p_{m+1} = p_{m+2} \ldots = p_n = 0$.

le - Limite de minutos extras que um motorista pode fazer por dia.

tt - Tempo de trabalho em uma jornada normal.

mindsc - Menor tempo de jornada após o qual se pode iniciar o descanso do motorista.

maxdsc - Maior tempo de jornada antes do qual se pode iniciar o descanso do motorista.

tdsc - Tempo mínimo de descanso.

tr - Tempo das rendições.

Variáveis

X - Para cada par de viagem i e motorista j, $X_{i,j} = 1$ sse a viagem i é realizada pelo motorista j.

X1 - $X1_{i,j} = 1$ sse a viagem i é coberta pelo motorista j antes do seu descanso.

X2 - $X2_{i,j} = 1$ sse a viagem i é coberta pelo motorista j após o seu descanso.

Enquanto as variáveis $X1$ e $X2$ devem ser declaradas binárias, X pode ser contínua em um intervalo entre 0 e 1 pois sua definição, explicada mais adiante ($X_{i,j} = X1_{i,j} + X2_{i,j}$), a leva a assumir somente os valores inteiros 0 e 1. Assim, o modelo ganha em eficiência.

I - I_j é o instante de início da jornada do motorista j.

D - D_j é o instante em que o motorista j começa seu descanso.

E - E_j é a quantidade de minutos extras trabalhados pelo motorista j e assume valores entre 0 e le.

As três últimas variáveis são declaradas contínuas, mas o modelo força para que elas sempre assumam valores inteiros na solução.

Restrições

São dadas a seguir as formulações matemáticas das restrições impostas ao escalonamento de motoristas. M é utilizado toda vez que se deseja expressar um inteiro positivo grande o suficiente, mas o valor de M em diferentes restrições nem sempre é o mesmo.

$p_j * M \geq \sum_{i \in PCs} X_{i,j}$ para $j = 1..n$

As viagens não podem ser atribuídas a motoristas que não estão disponíveis. Se $p_j = 1$, a restrição torna-se redundante, mas quando $p_j = 0$, todas as variáveis X relacionadas a j são zeradas, ou seja, nenhuma viagem é atribuída ao motorista j.

$v_i \geq (1 - X_{i,j}) * (-M) + I_j$ para $i \in PCs, j = 1..n$

Toda viagem atribuída a um motorista começa depois que sua jornada já foi iniciada. Quando $X_{i,j}$ vale 0, a restrição torna-se redundante. Porém, se $X_{i,j}$ vale 1, então a viagem i foi atribuída ao motorista j e o instante v_i de saída da viagem i deve ser posterior ao início da jornada do motorista j, I_j.

$v_i + d_i \leq (1 - X_{i,j}) * M + I_j + tt + E_j$ para $i \in PCs, j = 1..n$

Toda viagem atribuída a um motorista termina antes do fim de sua jornada. O tempo máximo de uma jornada é de tt minutos mais as horas extras. Portanto, quando $X_{i,j}$ vale 1, a viagem i foi atribuída ao motorista j e o instante do término da viagem i, $v_i + d_i$, deve ser anterior ao fim da jornada do motorista j, $I_j + tt + E_j$. Quando $X_{i,j}$ vale 0, a restrição torna-se redundante.

$v_i + d_i \leq (1 - X1_{i,j}) * M + D_j$ para $i \in PCs, j = 1..n$

Esta restrição define $X1$. Quando $X1_{i,j}$ vale 0, a restrição torna-se redundante. Porém, quando $X1_{i,j}$ vale 1, a viagem i termina antes do descanso do motorista j responsável pela viagem.

$v_i \geq (1 - X2_{i,j}) * (-M) + D_j + tdsc$ para $i \in PCs, j = 1..n$

Esta restrição define $X2$. Quando $X2_{i,j}$ vale 0, a restrição torna-se redundante. Se vale 1, é porque a viagem i começa depois de terminar o descanso do motorista j responsável pela viagem.

$X_{i,j} = X1_{i,j} + X2_{i,j}$ para $i \in PCs, j = 1..n$

A restrição acima determina X binária. Também faz com que $X1_{i,j}$ e $X2_{i,j}$ nunca sejam simultaneamente 1, ou seja, uma viagem não pode estar ao mesmo tempo antes e depois do descanso de um motorista.

$D_j \geq I_j + mindsc$ para $j = 1..n$

$D_j \leq I_j + maxdsc$ para $j = 1..n$

As duas restrições anteriores forçam o descanso dos motoristas a iniciar pelo menos $mindsc$ e no máximo $maxdsc$ minutos após a jornada ter começado.

$\sum_{j=1}^{n} X_{i,j} = 1$ para $i \in PCs$

Garantia de que a demanda é atendida.

$X_{i1,j} + X_{i2,j} \leq 1$ para $i1 \in PC1, i2 \in PC2, j = 1..n$ e $v_{i1} \leq v_{i2}; v_{i1} + d_{i1} > v_{i2}$

$X_{i1,j} + X_{i2,j} \leq 1$ para $i1 \in PC1, i2 \in PC2, j = 1..n$ e $v_{i2} \leq v_{i1}; v_{i2} + d_{i2} > v_{i1}$

$X_{i1,j} + X_{i2,j} \leq 1$ para $i1 \in PC1, i2 \in PC1, j = 1..n$ e $v_{i2} < v_{i1}; v_{i2} + d_{i2} + d2_{i2} > v_{i1}$

$X_{i1,j} + X_{i2,j} \leq 1$ para $i1 \in PC2, i2 \in PC2, j = 1..n$ e $v_{i2} < v_{i1}; v_{i2} + d_{i2} + d2_{i2} > v_{i1}$

As quatro restrições acima proíbem que viagens partam de um PC antes que o ônibus chegue nele e, também, que motoristas façam viagens sucessivas partindo de um mesmo PC, pois entre elas deve haver uma viagem partindo do PC oposto. Estas restrições não determinam a intercalação de viagens partindo ora do PC1, ora do PC2. No entanto, garantem que entre duas viagens partindo de um mesmo PC, sempre há tempo suficiente para uma viagem que parte do PC oposto. Assim, quando duas viagens que estão na tabela inicial de horários são sucessivas na jornada de um motorista e partem de um mesmo PC, uma viagem válida saindo do PC oposto deve ser acrescentada entre elas. Como foi citado na seção 4.1, a possibilidade de mudar a tabela inicial de horários permitiu uma flexibilidade maior na formulação do modelo, assegurando que a intercalação de viagens nos PCs fosse modelada dessa forma.

$X1_{i1,j} + X2_{i2,j} \leq 1$ para $i1 \in PC1, i2 \in PC1, j = 1..n$ e

$$v_{i2} > v_{i1}; v_{i2} \geq v_{i1} + d_{i1} + d2_{i1}; v_{i2} < v_{i1} + d_{i1} + d2_{i1} + tdsc$$

$X1_{i1,j} + X2_{i2,j} \leq 1 \quad$ para $i1 \in PC2, i2 \in PC2, j = 1..n$ e

$$v_{i2} > v_{i1}; v_{i2} \geq v_{i1} + d_{i1} + d2_{i1}; v_{i2} < v_{i1} + d_{i1} + d2_{i1} + tdsc$$

As duas últimas restrições garantem que, se uma viagem deve ser acrescentada à jornada de certo motorista, ela não deve coincidir com o seu período de descanso. Considere cada par de viagens em um mesmo PC ($i1$,$i2$) tal que $i2$ é posterior a $i1$ e há tempo para uma viagem entre $i1$ e $i2$ partindo do PC oposto, mas não há tempo para a viagem e o descanso. Neste caso, não pode ocorrer de, simultaneamente, $i1$ partir antes do descanso ($X1_{i1,j} = 1$) e $i2$ depois ($X2_{i2,j} = 1$) na jornada de um mesmo motorista, pois assim haveria um descanso entre $i1$ e $i2$, além de pelo menos uma viagem obrigatória.

Neste sistema, o escalonamento de motoristas é feito antes do escalonamento de ônibus. No entanto, estas duas etapas influem consideravelmente uma na outra. O início e o término de uma jornada, por exemplo, pode se dar, dependendo da escala de ônibus, tanto na garagem quanto em algum PC, quando ocorre rendição.

Suponha que apenas um motorista utilize determinado ônibus. Neste caso, além das viagens determinadas pela escala de motoristas, deve-se considerar as viagens de início e fim que envolvem a garagem. A adição dessas viagens à jornada do motorista poderia extrapolar o limite de tempo permitido, se elas não fossem previstas na escala de motoristas. Optou-se, então, por prever que, durante o escalonamento de motoristas, toda jornada começa e termina na garagem. Posteriormente, algumas destas viagens envolvendo a garagem podem ser eliminadas pelo escalonamento de ônibus quando a troca de motoristas pode ser feita em um PC. Dessa forma, garante-se a existência de pelo menos uma escala de veículos válida que satisfaça à escala de motoristas gerada.

A viagem final à garagem é considerada no modelo simplesmente substituindo a restrição

$$v_i + d_i \leq (1 - X_{i,j}) * M + I_j + tt + E_j \qquad \text{para } i \in PCs, j = 1..n$$

por

$$v_i + d_i + dgar2_i \leq (1 - X_{i,j}) * M + I_j + tt + E_j \qquad \text{para } i \in PCs, j = 1..n$$

E a viagem inicial que parte da garagem é prevista com a substituição de

$$v_i \geq (1 - X_{i,j}) * (-M) + I_j \qquad \text{para } i \in PCs, j = 1..n$$

por

$$v_i \geq (1 - X_{i,j}) * (-M) + I_j + dgar_i \qquad \text{para } i \in PCs, j = 1..n$$

O modelo descrito até aqui é denominado *modelo base* porque os resolvedores de programação por restrições e de programação linear o utilizam como ponto de partida, e restrições podem ser adicionadas ou removidas durante o processo de escalonamento de motoristas.

O algoritmo que resolve o PPV utiliza o modelo primeiramente com $p_1 = 1$ e $p_2 = p_3 = \ldots = p_n = 0$. Se não encontra solução, tenta com $p_1 = p_2 = 1$ e $p_3 = p_4 = \ldots = p_n = 0$ e assim por diante, até encontrar uma solução. Antes de expor o algoritmo completo, é necessário identificar a simetria no modelo base.

4.2.2 Simetria

Um modelo apresenta simetria quando vários pontos do espaço de busca representam uma mesma solução. Isso implica em modelos desnecessariamente grandes, uma vez que certas combinações são equivalentes e basta que apenas uma delas seja testada.

Simetrias, que são uma dificuldade para a programação inteira, podem ser eficientemente controladas por técnicas de programação por restrições. Em [49], identifica-se três maneiras de se reduzir simetria em modelos de programação por restrições: reformular o problema em um modelo com menos simetrias, adicionar restrições que quebram a simetria do modelo, e adicionar restrições durante a busca [25] de forma que soluções simétricas não sejam exploradas. As duas primeiras técnicas também podem ser usadas na programação matemática, mas a última só é possível na programação por restrições.

A simetria está presente no modelo proposto para resolver o problema de escalonamento de motoristas e é fácil identificá-la através de um exemplo. O modelo não permite que dois motoristas cubram uma mesma viagem, portanto, não há como uma jornada de trabalho ser igual a outra já que cada viagem deve estar presente em uma única jornada. Considere duas soluções distintas A e B e os motoristas indexados de 1 a n. Suponha que todas as jornadas dos motoristas, exceto de 1 e de 2, sejam idênticas nas soluções A e B. Se a jornada do motorista 1 da solução A é idêntica à jornada do motorista 2 da solução B, e a jornada do motorista 2 da solução A é idêntica à jornada do motorista 1 da solução B, então temos duas soluções simétricas pois representam a mesma escala.

Uma maneira de se garantir que as simetrias sejam evitadas é ordenar os motoristas de acordo com sua viagem inicial. Considere a *viagem inicial* de um motorista a primeira viagem feita por ele entre dois PCs. Note que, se o motorista começa sua jornada saindo da garagem, esta não é considerada sua viagem inicial pois não envolve dois PCs. Se garantirmos que a viagem inicial do motorista j vem antes da viagem inicial do motorista $j + 1$, então apenas uma das soluções simétricas é verificada.

Como é bem mais complicado eliminar simetrias através de restrições na forma de expressões lineares no modelo base, optou-se por eliminar a simetria do problema durante a busca de soluções no modelo de programação por restrições, que também utiliza o modelo base como ponto de partida.

No entanto, percorrer todas as soluções ainda é muito dispendioso. Uma alternativa seria deixar para a programação por restrições encontrar as possíveis combinações de

viagens iniciais e, para cada combinação, encontrar a solução que minimiza a quantidade de horas extras através da programação matemática. Desta forma, obtém-se a solução ótima para cada conjunto possível de viagens iniciais e a melhor solução entre elas é a que apresenta o melhor valor para a função objetivo. O trecho de código 4.2 mostra as mudanças que devem ser feitas na primeira cláusula de *instancia/2* do algoritmo 4.1.

```
instancia(Vars,QtdeMot):-
    retract_all(melhor_solucao(_,_)), // exclui fatos melhor_solucao/2.
    labeling(Vars,QtdeMot,ViagensIniciais), // retorna as viagens iniciais
    proibe_viagens_iniciais(ViagensIniciais),
    // ViagensIniciais não pode mais ser retornado como resposta de labeling/3
    modelo_PLI(QtdeMot,ViagensIniciais,Solucao),
    // retorna a solução ótima do modelo para o conjunto de viagens iniciais fornecido
    custo(Solucao,Custo), // calcula o custo de uma instanciação
    (melhor_solucao(_,MelhorCusto), Custo ≤ MelhorCusto → // '→'=então
        true; // ';'=senão
        retract_all(melhor_solucao(_,_)),
        assert(melhor_solucao(Solucao,Custo)) // inclui fato melhor_solução/2.
    ),
    fail. // Esta falha faz com que todas as instanciações sejam testadas
```

Algoritmo 4.2: Interação entre os modelos PR e PLI

Assim, o algoritmo extrai as vantagens da programação inteira e da programação por restrições, combinando a facilidade de lidar com simetrias da primeira técnica com a eficiência na descoberta de soluções ótimas da segunda. O modelo de programação linear inteira é explicado na seção 4.2.4

A otimalidade da solução estaria provada se todas as combinações de viagens iniciais fossem testadas, pois trata-se de um algoritmo de busca completo. Porém, para fazer com que o programa termine em tempo razoável, foram feitas algumas alterações, explicadas adiante, que, apesar de impedirem a prova da otimalidade da solução, parecem não comprometer sua qualidade. É possível, por exemplo, limitar a quantidade de conjuntos diferentes de viagens iniciais a serem utilizados no modelo de programação inteira sem degradar a solução de forma considerável.

A seguir são explicados a estratégia de busca utilizada na programação por restrições, o modelo de programação linear inteira e os mecanismos utilizados para permitir que o programa termine em tempo razoável.

4.2.3 Estratégia de Busca

Conforme [47], os métodos de busca diferenciam-se na maneira como percorrem o espaço de busca, segundo os seguintes critérios:

Exploração Completa x Incompleta - A busca completa se dá quando o espaço de busca é percorrido de tal forma que todas as soluções são garantidamente encontradas. Neste caso, é possível encontrar a solução ótima. Já a busca incompleta deve ser suficiente para encontrar uma solução adequada.

Construtivo x Busca Local - Tem-se uma *atribuição total* quando todas as variáveis de decisão assumem um valor definido. Uma *atribuição parcial*, por sua vez, considera que apenas um subconjunto das variáveis de decisão estão instanciadas. Em métodos construtivos, as atribuições parciais são estendidas até que se encontre atribuições totais, que correspondem a soluções. Em métodos de busca local, as soluções são encontradas movendo-se de uma atribuição total para outra.

Aleatoriedade - Alguns métodos usam mecanismos aleatórios enquanto outros seguem sempre regras fixas.

Se é possível fazer uma exploração completa do espaço de busca em tempo razoável, os métodos construtivos são os mais indicados e não há necessidade de se usar métodos aleatórios. Como o objetivo é encontrar a solução ótima para o PPV, testou-se um método de busca construtivo, sem elementos aleatórios e que faz exploração completa.

A busca em árvore pode ser utilizada neste caso. Uma forma especialmente econômica é a busca em profundidade da esquerda para a direita por *backtracking*, pois os recursos de memória necessários são de ordem linear no tamanho da entrada, ao contrário de outros algoritmos de *backtracking* [2].

No PPV, não há o risco de a busca em profundidade entrar em um ramo infinito da árvore pois ela é finita. Além disso, uma certa flexibilidade pode ser explorada com diferentes estratégias de seleção de variáveis e de valores.

Ordenação das variáveis

A ordem em que as variáveis são instanciadas influi de maneira considerável na quantidade de falhas que ocorrem durante o *backtracking* antes de se encontrar uma solução viável ou a solução ótima. Como existem muitas formas de se percorrer o espaço de busca, essa ordem deve considerar o domínio das variáveis e a maneira com que elas se relacionam às outras variáveis através das restrições, o que torna a eficiência da busca muito dependente das

características do problema. Os métodos de busca dividem-se em dinâmicos e estáticos. Enquanto nestes a ordem das variáveis é decidida no início da busca e se mantém até o fim, naqueles a ordem pode ser alterada durante a busca, com o ônus de se guardar informações extras.

A ordenação de variáveis utilizada no algoritmo proposto é estática pois isso permite maior controle nos cortes do espaço de busca e na adição de restrições durante a busca, o que seria mais difícil se a ordem das variáveis fosse mutável. Além disso, a ordenação estática exige menos memória e é mais simples de se entender e implementar.

As variáveis de decisão X podem ser ordenadas conforme o momento de partida da viagem e o motorista a que se referem. Porém, como uma permutação entre os motoristas não altera a solução do modelo, a ordem imposta a eles não afeta a busca.

No que diz respeito às viagens, as heurísticas e os procedimentos manuais normalmente consideram as viagens na ordem cronológica crescente dos momentos de partida e é esta ordenação que será utilizada. Assim, as variáveis são instanciadas na seqüência $X_{1,1}, X_{1,2}, \ldots, X_{1,n}, X_{2,1}, X_{2,2}, \ldots, X_{2,n}, \ldots, X_{v,1}, X_{v,2}, \ldots, X_{v,n}$, em que v corresponde à última viagem da tabela inicial de horários e n à quantidade de motoristas disponíveis.

As variáveis também poderiam ser instanciadas na seqüência $X_{1,1}, X_{2,1}, \ldots, X_{v,1}, X_{1,2}$, $X_{2,2}, \ldots, X_{v,2}, \ldots, X_{1,n}, X_{2,n}, \ldots, X_{v,n}$, mas experimentos mostraram que a ordem anterior resulta em maior eficiência.

Ordenação de valores

As variáveis de decisão X são as únicas instanciadas no modelo de programação por restrições. As demais são instanciadas de acordo com as soluções dos modelos de programação inteira ao final do algoritmo.

Durante a busca, o primeiro valor tentado para qualquer variável X é um e, se há falha, tenta-se zero. Isso porque a atribuição do valor um a uma variável X neste modelo gera muito mais propagação de restrições do que a atribuição do valor zero. Esta ordem também facilita a quebra de simetria e o corte no espaço de busca, pois, combinada com uma ordenação de variáveis em que são considerados primeiro os motoristas correspondentes aos menores índices, permite que os motoristas de menor índice sempre iniciem suas jornadas mais cedo.

Ordenação de variáveis e valores

Foram citadas duas ordenações de variáveis e duas de valores. As quatro diferentes combinações foram testadas e a que apresentou maior eficiência utilizou a seqüência de valores [1,0] e de variáveis $X_{1,1}, X_{1,2}, \ldots, X_{1,n}, X_{2,1}, X_{2,2}, \ldots, X_{2,n}, \ldots, X_{v,1}, X_{v,2}, \ldots, X_{v,n}$.

Note que, percorrendo o espaço de busca desta forma, procura-se atribuir primeiramente uma viagem ao motorista 1 ($X_{i,1} = 1$). Se não for possível, tenta-se atribui-la ao motorista 2 ($X_{i,2} = 1$) e assim por diante, até que seja atribuída a um motorista. Assim, um motorista só inicia sua jornada quando uma viagem não pode ser coberta por nenhum dos outros motoristas que já começaram sua jornada.

Algoritmo de busca

O algoritmo de busca 4.3 faz a instanciação de uma lista de variáveis cujo próximo elemento a ser instanciado é $X_{v,j}$ e pode-se utilizar até *QtdeMot* motoristas. Seu principal objetivo, porém, é retornar um conjunto de viagens iniciais, *ViagensIniciais*, para ser utilizado pelo modelo PLI, como já foi mostrado no algoritmo 4.2.

labeling([ ],QtdeMot,ViagensIniciais):-
    retorne o conjunto de viagens iniciais em ViagensIniciais.
labeling([$X_{v,j}$|Y],QtdeMot,ViagensIniciais):-
    1. Se $1 \in D(X_{v,j})$ então // *D($X_{v,j}$)=domínio de* $X_{v,j}$
        2. Se alguma viagem já foi atribuída a j então
            indomain($X_{v,j}$),
            labeling(Y,QtdeMot,ViagensIniciais)
        2. senão
            3. Se próximo_motorista_a_sair=j e $j \leq$ QtdeMot então
                $X_{v,j}$:=1,
                Guarde que $X_{v,j}$ é a viagem inicial de j
                próximo_motorista_a_sair=$j+1$
                labeling(Y,QtdeMot,ViagensIniciais),
            3. senão
                $X_{v,j}$:=0,
                labeling(Y,QtdeMot,ViagensIniciais),
    1. senão
        $X_{v,j}$:=0,
        labeling(Y,QtdeMot,ViagensIniciais)

Algoritmo 4.3: Algoritmo de busca

1. Quando o algoritmo é iniciado, todas as variáveis X podem assumir os valores 0 ou 1. No entanto, devido à propagação de restrições, alguns domínios podem ser reduzidos. Se a próxima variável $X_{v,j}$ a ser instanciada tem seu domínio vazio, a busca falha e ocorre o *backtracking*. Se o domínio contém somente zero, este valor é assumido e a instanciação das outras variáveis continua. Porém, se o domínio contém o valor 1, então o passo 2 é executado.

2. Se o motorista j já cobre viagens anteriores a v, instancia-se $X_{v,j}$ primeiramente com o valor 1, e depois com 0, caso ocorra *backtracking*. Se o motorista j ainda não começou sua jornada, o passo 3 é executado.

3. Se os motoristas $1 \ldots j-1$ já começaram suas jornadas e j não, atribui-se um a $X_{v,j}$ e elimina-se o ponto de escolha em que $X_{v,j}$ assumiria o valor zero, evitando assim soluções simétricas. Agora, o próximo motorista a iniciar sua jornada é $j+1$. No entanto, se j não é o próximo motorista a sair ou se a inclusão de j exige uma quantidade de motoristas maior do que a disponível, v não é atribuída a j ($X_{v,j} = 0$).

A instanciação através do predicado *indomain* funciona da seguinte maneira. Se, no momento em que é instanciada, o domínio da variável é vazio, a busca falha e ocorre *backtracking*. Se o domínio é unitário, a variável assume o único valor possível. Se é binário, primeiramente um valor é atribuído e, na seqüência, caso a busca falhe, o outro valor é atribuído.

O operador ':=' representa os casos em que apenas um valor é utilizado. Nos casos em que o zero é usado ($X_{v,j} := 0$), não há corte na árvore de busca pois são situações em que o valor 1 já não pertence ao domínio da variável. No entanto, quando a $X_{v,j}$ é atribuído o valor 1 ($X_{v,j} := 1$), há corte no ramo da árvore em que $X_{v,j} = 0$. Portanto, o único passo do algoritmo onde pontos de escolha podem ser criados é na condição 2, através do predicado *indomain*.

Observando-se o comportamento do algoritmo 4.3, percebe-se que ele realiza muitos passos de *backtracking* quando trata as variáveis relativas a viagens que não são iniciais. Isso porque procura solução para todas as variáveis X. No entanto, estes valores não fazem nenhuma diferença ao modelo PLI, que se importa apenas com variáveis referentes às viagens iniciais. Além disso, a verificação de existência de uma solução para todas as variáveis X já é feita naturalmente pelo próprio modelo PLI.

Então, para melhorar o algoritmo de busca, basta acrescentar um teste que retorna o conjunto de viagens iniciais quando todos os motoristas disponíveis já estiverem com pelo menos uma viagem atribuída. Desta forma, o predicado representado em 4.4 é finalizado em duas situações: quando todas as variáveis X tiverem sido instanciadas, assim como em 4.3, ou quando a viagem inicial de cada motorista disponível já estiver estabelecida.

4.2.4 Modelo de Programação Linear Inteira

O modelo PLI é formado pelo modelo base acrescido de algumas restrições. Para gerá-lo são necessários os seguintes dados e restrições complementares:
nmax - Quantidade máxima de motoristas que pode ser utilizada na linha.
n - Quantidade de motoristas disponíveis no teste corrente.

```
labeling([ ],QtdeMot,ViagensIniciais):-
    retorne o conjunto de viagens iniciais em ViagensIniciais.
labeling([X_{v,j}|Y],QtdeMot,ViagensIniciais):-
    1. Se 1 ∈ D(X_{v,j}) então        // D(X_{v,j})=domínio de X_{v,j}
       2. Se alguma viagem já foi atribuída a j então
          indomain(X_{v,j}),
          labeling(Y,QtdeMot,ViagensIniciais)
       2. senão
          3. Se próximo_motorista_a_sair=j então
             X_{v,j}:=1,
             Guarde que X_{v,j} é a viagem inicial de j,
             4. Se não há mais motoristas a sair (j=QtdeMot) então
                retorne o conjunto de viagens iniciais em ViagensIniciais
             4. senão
                próximo_motorista_a_sair=j+1,
                labeling(Y,QtdeMot,ViagensIniciais)
          3. senão
             X_{v,j}:=0,
             labeling(Y,QtdeMot,ViagensIniciais)
    1. senão
       X_{v,j}:=0,
       labeling(Y,QtdeMot,ViagensIniciais)
```

Algoritmo 4.4: Busca melhorada

ViagensIniciais - Conjunto de pares (v, j) em que v corresponde à viagem inicial do motorista j.
c - Custo da melhor solução encontrada para a instanciação parcial corrente, até o instante em que o modelo é construído.

A seguinte função objetivo é usada:

$$minimize \sum_{j=1}^{n} E_j$$

O objetivo é minimizar a quantidade de horas extras.

As restrições adicionais são as seguintes:

$$X_{v,j} = 1 \quad \text{para } (v,j) \in ViagensIniciais$$

$$X_{z,j} = 0 \quad \text{para } (v,j) \in ViagensIniciais \text{ e } z < v$$

As duas restrições anteriores determinam quais são as viagens iniciais de cada motorista. A primeira diz que a viagem inicial v de um motorista j deve estar em j e a segunda estabelece que qualquer viagem anterior à v não pode estar em j.

$\sum_{j=1}^{n} E_j \leq c - 1$

Antes do modelo PLI ser ativado para determinado conjunto de viagens iniciais, é conhecido o custo c da melhor solução corrente. Assim, em vez de se obter a solução ótima do modelo e compará-la a c, impôs-se a restrição acima, tornando o modelo mais eficiente, o que foi confirmado pelos experimentos. Caso alguma solução seja alcançada, ela passa a ser a melhor dentre as combinações de viagens iniciais já testadas. Com o uso desta restrição, o trecho de código em 4.2 é levemente alterado mas mantém-se o princípio de seu funcionamento.

$p_j = 1 \quad$ para $j = 1..n$

$p_j = 0 \quad$ para $j = n + 1..nmax$

Os parâmetros p do modelo PLI devem herdar os valores de p do modelo PR.

A maioria dos modelos de programação inteira gerados pelo sistema são resolvidos rapidamente. No entanto, alguns podem demorar mais de uma hora para retornar uma resposta, seja ela uma solução ou a confirmação de que não há solução. Torna-se necessário, então, considerar alguma forma de se limitar o tempo utilizado na resolução desses modelos, o que pode ser conseguido configurando-se um parâmetro. Quando este limite vence, o resolvedor pode ter encontrado alguma solução que, apesar de geralmente não corresponder à ótima, representa um ganho em relação à melhor alcançada até o momento.

4.2.5 Instanciações Parciais

Se todas as combinações possíveis de viagens iniciais fossem consideradas, haveria uma explosão combinatória no número de soluções viáveis, mesmo em instâncias não muito grandes do problema. A maneira encontrada para transpor esta dificuldade foi dividir a instanciação das variáveis em etapas. A cada uma destas etapas damos o nome de *instanciação parcial*, que é determinada pela quantidade de variáveis que devem ser instanciadas e de motoristas que podem ser utilizados para efetuar as viagens correspondentes a estas variáveis.

O algoritmo 4.5 instancia um conjunto de variáveis, atribuindo as viagens relacionadas a elas a um conjunto restrito de motoristas. Em seguida, instancia outro conjunto de variáveis da mesma forma e assim por diante, até que todas as variáveis tenham um valor associado.

O critério utilizado para determinar as instanciações parciais explora as características do problema. Normalmente, a demanda pelos serviços de ônibus urbano apresenta dois horários de pico, um pela manhã (PM) e outro à tarde (PT), como mostra a figura 4.2. Para atender à demanda, é necessário lançar mão de muitos ônibus nos momentos que antecedem os picos. Isso faz com que o começo da jornada de muitos motoristas se dê em instantes próximos. Os motoristas destacados para cobrir os picos podem ser usados também para efetuar grande parte das viagens que não partem nos horários de pico. Assim, para que a demanda de todas as faixas horárias sejam atendidas, poucos motoristas precisam ser acrescentados àqueles que já efetuam as viagens dos picos.

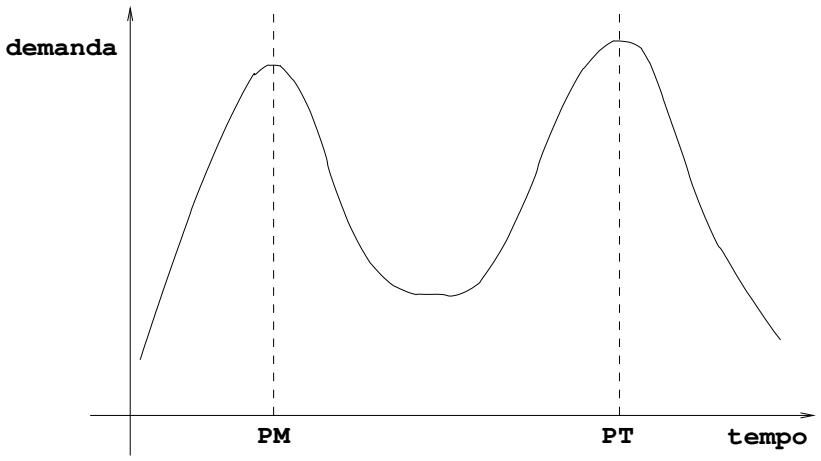


Figura 4.2: Curva de demanda típica ao longo do dia

O procedimento que determina as instanciações parciais percorre normalmente o espaço de busca, mas quando uma viagem inicial v demora muito a sair em relação à viagem inicial anterior, o algoritmo pára, determinando que as variáveis a serem instanciadas são aquelas relacionadas às viagens anteriores a v, e os motoristas a serem utilizados são os que foram necessários nesta pré-instanciação.

O funcionamento do algoritmo é explicado através de um exemplo em que a seqüência $v_1, v_2, \ldots, v_{12}, \ldots, v_n$ corresponde às viagens da linha. Durante a busca, determina-se que v_1, v_2, v_4 e v_6 são, respectivamente, viagens iniciais dos motoristas 1, 2, 3 e 4, e todas as outras viagens até v_{11} são cobertas por estes mesmos motoristas. Se v_{12} não pode ser efetuada por nenhum destes motoristas, um novo deve ser alocado. No entanto, a última viagem inicial anterior a v_{12} foi v_6, a seis unidades de distância, e, se 5 unidades já

```
busca(Vars,QtdeMot):-                  // No algoritmo 4.1 em busca/2,
    instancia_tudo(Vars,QtdeMot), !. // substitui instancia/2 por instancia_tudo/2

instancia_tudo([ ],_):-!.
instancia_tudo(Vars,QtdeMot):-
    pre_labeling(Vars,QtdeMot,InstanciacaoParcial),   // determina a instanciação parcial
    InstanciacaoParcial=(VarsParcial,QtdeMotParcial),
    instancia(VarsParcial,QtdeMotParcial), // predicado definido nos algoritmos 4.1 e 4.2
    RestoVars=Vars - VarsParcial,
    instancia_tudo(RestoVars,QtdeMot),
    !.
```

Algoritmo 4.5: Instanciações parciais

representam demora - este valor é passado como parâmetro ao sistema[1] - a primeira instanciação parcial é determinada pelas variáveis das viagens $v_1, \ldots, v_{11}$ e por 4 motoristas disponíveis.

Suponha agora que as possíveis combinações de viagens iniciais para cobrir estas 11 viagens com 4 motoristas sejam $\{[v_1,v_2,v_4,v_6],[v_1,v_2,v_5,v_6],[v_1,v_2,v_3,v_4]\}$. Este conjunto é descoberto pelo predicado *labeling/3* definido no algoritmo 4.4 e chamado em 4.2.

Para cada combinação, constrói-se um modelo PLI baseado nas restrições explicadas na seção 4.2.4, e com a supressão de algumas restrições referentes à demanda. A expressão

$\sum_{j=1}^{n} X_{i,j} = 1 \qquad$ para $i \in PCs$

é substituída por

$\sum_{j=1}^{n} X_{i,j} = 1 \qquad$ para $i \in PCs$ e $i \leq k$,

onde k é a quantidade de viagens que devem ser cobertas na instanciação parcial corrente. Na última instanciação parcial, em que todas as variáveis terminarão instanciadas, a expressão utilizada é equivalente à original pois k corresponde à última viagem. No exemplo citado, k vale 11.

O algoritmo 4.5 fixa todas as variáveis envolvidas em uma instanciação parcial de forma que as etapas posteriores não podem modificar seus valores. Assim, cada instanciação parcial influi de maneira decisiva nas instanciações futuras pois, naturalmente, quanto mais variáveis forem fixadas, menos combinações são possíveis para instanciações posteriores, tornando mais eficiente a resolução do problema. Portanto, deve haver um compromisso entre quais variáveis devem ser fixadas ao final de cada instanciação parcial e o tempo máximo permitido para resolver o problema. Em outras palavras, um compromisso entre a qualidade da solução e a eficiência do programa.

Podem ser adotadas outras estratégias que não fixam os valores de todas as variáveis envolvidas em determinada instanciação parcial com o uso de *desfaz_instanciacoes/1* e

[1] Em todos os testes realizados, este parâmetro assumiu o valor 5.

fixa_variaveis/1. Aquele torna livre as variáveis passadas como parâmetro e este funciona como uma forma de imunizar as variáveis contra o primeiro. Assim, uma vez que certa variável passa pela condição de *fixa_variaveis/1*, ela não pode mais tornar-se livre através de *desfaz_instanciacoes/1*. Nesta abordagem, representada pelo algoritmo 4.6, *desfaz_instanciacoes/1* é aplicada a todas as variáveis, tornando-as livres dos valores atribuídos por *instancia/2*, exceto as que satisfazem às condições de *fixa_variaveis/1*.

```
instancia_tudo(Vars,QtdeMot):-
    instancia_tudo(Vars,Vars,QtdeMot).
instancia_tudo([ ],_,_):-!.
instancia_tudo(Vars,VarsLivres,QtdeMot):-
    pre_labeling(VarsLivres,QtdeMot,InstanciacaoParcial), //determina a instanciação parcial
    desfaz_instanciacoes(Vars),
    InstanciacaoParcial=(VarsParcial,QtdeMotParcial),
    instancia(VarsParcial,QtdeMotParcial), // predicado definido nos algoritmos 4.1 e 4.2
    fixa_variaveis(VarsParcial),
    RestoVars=VarsLivres - VarsParcial,
    instancia_tudo(Vars,RestoVars,QtdeMot),
    !.
```

Algoritmo 4.6: Introdução de *desfaz_instanciacoes/1* e *fixa_variaveis/1*

Repare que o novo algoritmo pode ser definido para funcionar como seu antecessor, simplesmente fazendo com que *fixa_variaveis/1* amarre os valores de todas as variáveis da instanciação parcial corrente. No sistema implementado fixa-se apenas as viagens iniciais descobertas pela instanciação parcial. Desta forma, em relação a 4.5, o algoritmo 4.6 não apresenta grande perda de eficiência e a solução encontrada normalmente é bem melhor.

4.2.6 Eficiência

Alguns mecanismos, como a utilização de instanciações parciais, foram criados para tornar o programa mais eficiente. Restringir a quantidade de conjuntos de viagens iniciais testados pelo modelo PLI é também uma maneira de melhorar o desempenho do sistema. Assim, utiliza-se apenas um subconjunto das combinações possíveis de viagens iniciais, cujo tamanho é determinado pelo usuário através de um parâmetro.

Juntos, os algoritmos 4.2 e 4.4 implementam uma busca completa nas soluções do escalonamento de motoristas, mas respostas em tempo razoável são possíveis apenas para instâncias pequenas. Para sanar este problema, foram criados os seguintes mecanismos, já explicados:

- Utilização de instanciações parciais.

- Limite no tempo para resolver os modelos de programação linear inteira.
- Limite na quantidade de combinações de viagens iniciais testadas.

Como estes mecanismos tornam a busca incompleta, uma solução não pode ser provada ótima. Por outro lado, há um ganho enorme em eficiência, sem grande perda na qualidade da solução, conforme mostraram os experimentos.

4.3 Escalonamento de Veículos

O escalonamento de motoristas gera uma solução que corresponde a um conjunto de jornadas de motoristas. Seguindo a figura 4.1, após esta etapa, deve ser realizado o escalonamento de ônibus, cujo principal objetivo é utilizar o mínimo de veículos para cobrir estas jornadas. Como as viagens envolvendo a garagem não são previstas no escalonamento de motoristas, as jornadas ainda estão incompletas antes de se iniciar o escalonamento de ônibus. Esta etapa, portanto, também tem a responsabilidade de completar as jornadas dos motoristas.

A solução do escalonamento de motoristas pode trazer jornadas em que o descanso vem antes mesmo da viagem inicial ou depois de todas as viagens, como ilustra a figura 4.3. Isso ocorre porque não há restrições que impeçam a construção de jornadas com estas configurações. Nestes casos, as jornadas, que a princípio eram padrões, transformam-se em contínuas ou, quando possível, são juntadas para comporem duplas pegadas. Se a formação de cada dupla pegada elimina a necessidade de um motorista, é interessante maximizar o uso deste tipo de jornada. Os três diferentes tipos de jornadas foram explicados no início da seção 4.2.

O algoritmo, então, procura utilizar o mínimo de veículos para cobrir as jornadas dos motoristas e o mínimo de horas extras para completá-las. Procura ainda, realizar o maior número de ligações entre jornadas de forma a torná-las duplas pegadas, reduzindo assim, a quantidade de motoristas necessários. Estes objetivos coincidem com aqueles identificados no início do capítulo 4.

Normalmente, no escalonamento de ônibus, juntam-se viagens. Nesse trabalho, ligam-se jornadas, o que torna o escalonamento de ônibus bem mais simples, já que manipula menos combinações. Para que dois motoristas utilizem um mesmo ônibus, a jornada do primeiro deve terminar no mesmo local e antes do momento em que começa a jornada do segundo. Seguindo essa intuição, o modelo do problema utiliza uma representação através de arcos. Para cada jornada resultante do escalonamento de motoristas são considerados 7 arcos, como mostra a figura 4.4:

- Um arco que representa a jornada como ela foi gerada pelo escalonamento de motoristas.

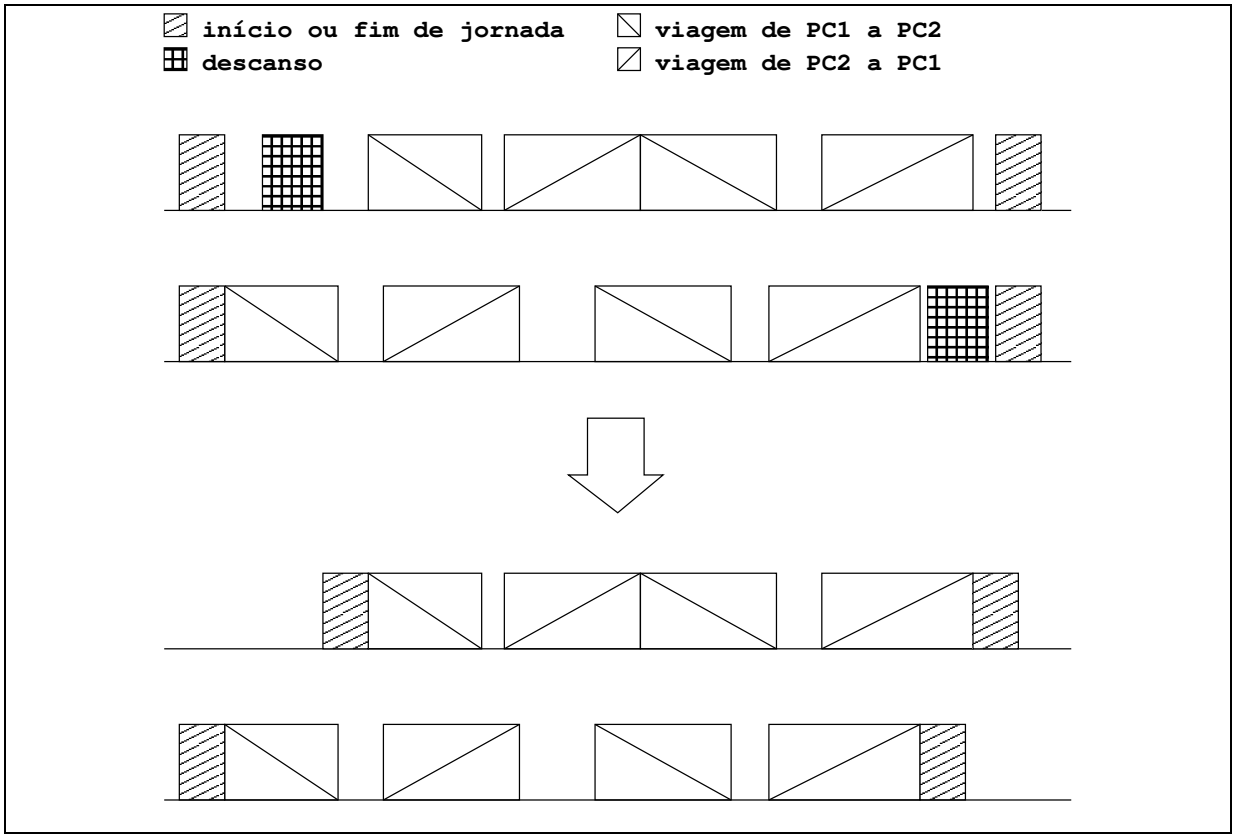


Figura 4.3: Transformação de jornada padrão para contínua ou turno de dupla pegada

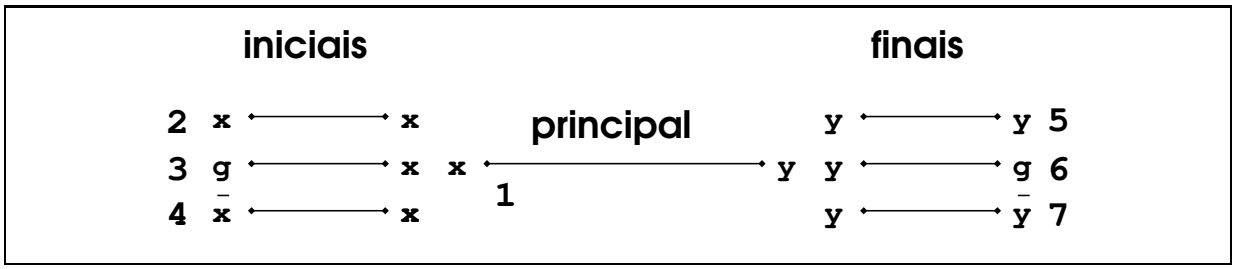


Figura 4.4: Arcos no escalonamento de veículos

- Três arcos que representam o início da jornada, pois ela pode começar a partir de três lugares distintos: da garagem (g), do próprio PC de onde sai a primeira viagem (x), ou do outro PC ($\bar{x}$). Deve haver um arco diferente para cada situação pois a duração da primeira atividade do motorista varia de acordo com a localidade de onde ela parte. O arco 2, por exemplo, tem valor zero, pois nenhuma viagem precisa ser adicionada para a jornada representada na figura sair do PC x.

- Três arcos que representam o fim da jornada, cuja indicação é similar àquela dos três arcos criados para a atividade inicial.

Estes arcos, indexados de 1 a 7 conforme a figura 4.4, são classificados respectivamente em principal, iniciais e finais. Para cada jornada de motorista deve ser selecionado um arco de cada tipo. O modelo, detalhado a seguir, foi construído para ser resolvido por programação matemática pelos seguintes motivos:

- É fácil representar o problema através de expressões lineares.
- A formulação é rápida quando se usa a linguagem de modelagem AMPL.
- O resolvedor é eficiente na busca de solução para o modelo.

Parâmetros

n - Define a quantidade de jornadas de motorista, determinada pelo escalonamento de motoristas.
le - Limite de minutos extras que um motorista pode fazer por dia.
tt - Tempo de trabalho em uma jornada normal.
mindsc - Menor tempo de jornada após o qual se pode iniciar o descanso do motorista.
maxdsc - Maior tempo de jornada antes do qual se pode iniciar o descanso do motorista.
tdsc - Tempo mínimo de descanso.
tr - Tempo de rendição.
ndp - Porcentagem máxima de jornadas do tipo dupla pegada que pode haver na escala de motoristas.
intdp - Intervalo mínimo entre os turnos de cada jornada dupla pegada.
tdp - Tamanho máximo de um turno da jornada dupla pegada.
co, **cm** e **ce** - Respectivamente, os custos da utilização de cada ônibus, motorista e minuto extra.
i e **f** - Cada arco determinado pela jornada do motorista j e pelo tipo $t \in 1..7$ tem um momento de início $i_{j,t}$ e de fim $f_{j,t}$.
pci e **pcf** - Cada arco determinado pela jornada do motorista j e pelo tipo $t \in 1..7$ é associado a um PC de início $pci_{j,t}$ e a outro de fim $pcf_{j,t}$.
dsc - A solução do escalonamento de motoristas pode trazer jornadas em que o descanso vem antes mesmo da viagem inicial ou depois de todas as viagens. No escalonamento de ônibus, estas jornadas são consideradas do tipo contínua ou turnos de duplas pegadas e devem vir com o parâmetro $dsc_j = 0$. Para as jornadas padrões, dsc_j corresponde ao momento em que se inicia o descanso do motorista.
dgar - Para cada motorista j, sabe-se o tempo $dgar_j$ que um ônibus leva para ir da garagem ao PC onde começa a jornada de j.
dgar2 - Para cada motorista j, sabe-se o tempo $dgar2_j$ que um ônibus leva para ir do PC onde termina a jornada de j à garagem.

Variáveis

DP - $DP_{j,k} = 1$ sse as jornadas dos motoristas j e k formam uma dupla pegada, sendo que o turno j é anterior ao k.
A - $A_{j,t} = 1$ sse o arco do tipo $t \in 1..7$ faz parte da jornada do motorista j.
Y1 - $Y1_j = 1$ sse o motorista j é o primeiro no dia a utilizar certo ônibus.
Y2 - $Y2_j = 1$ sse o motorista j é o segundo no dia a utilizar certo ônibus.
X - $X_{j,k} = 1$ sse os motoristas j e k utilizam o mesmo veículo, j antes de k.
I - I_j é o instante de início da jornada do motorista j.
F - F_j é o instante final da jornada do motorista j.
E - E_j é a quantidade de minutos extras trabalhados pelo motorista j e assume valores entre 0 e *le*.

As três últimas variáveis são contínuas e as demais são binárias.

Restrições

Novamente, M é utilizado para representar um número positivo suficientemente grande. Devido às características das linhas de ônibus testadas, convencionou-se que cada veículo é utilizado no máximo por dois motoristas.

$$X_{j,k} = 0 \qquad \text{para } j = 1..n, k = 1..n \text{ e } f_{j,1} > i_{k,1}$$

Um mesmo veículo não pode ser utilizado por dois motoristas que, em algum momento, cumprem suas jornadas simultaneamente.

$$\sum_{k=1}^{n} X_{j,k} - Y1_j \leq 0 \qquad \text{para } j = 1..n$$

$$1 - Y1_j \leq \sum_{k=1}^{n} (X_{j,k} + X_{k,j}) \qquad \text{para } j = 1..n$$

As duas restrições acima definem as variáveis $Y1$. A primeira estabelece $Y1_j = 1$ quando j é o primeiro motorista a conduzir um veículo que é utilizado por dois motoristas, e a segunda estabelece $Y1_j = 1$ quando j é o único motorista a utilizar determinado veículo.

$$\sum_{j=1}^{n} X_{j,k} - Y2_k \leq 0 \qquad \text{para } k = 1..n$$

Restrição principal para as variáveis $Y2$.

$Y1_j + Y2_j = 1 \qquad \text{para } j = 1..n$

Garantia de que um motorista seja o primeiro ou o segundo a conduzir algum veículo, mas não ambos.

$A_{j,1} = 1 \qquad \text{para } j = 1..n$

$\sum_{t=2}^{4} A_{j,t} = 1 \qquad \text{para } j = 1..n$

$\sum_{t=5}^{7} A_{j,t} = 1 \qquad \text{para } j = 1..n$

As restrições acima determinam que, para cada motorista presente na solução do escalonamento de motoristas, deve ser selecionado um arco de cada tipo: principal, de início e de fim.

$Y1_j \leq A_{j,3} \qquad \text{para } j = 1..n$

Todos os motoristas que são os primeiros a conduzir os respectivos veículos começam suas jornadas da garagem.

$Y2_j \leq A_{j,6} \qquad \text{para } j = 1..n$

Todos os motoristas que são os segundos de seus respectivos veículos terminam suas jornadas na garagem, pois convencionou-se que no máximo dois motoristas podem usar um mesmo ônibus.

$A_{j,6} \geq 1 - \sum_{k=1}^{n} X_{j,k} \qquad \text{para } j = 1..n$

Devem também terminar na garagem as jornadas dos motoristas que são os únicos a conduzir determinado ônibus.

$X_{j,k} + A_{j,t} - A_{k,s} \leq 1 \qquad \text{para } j = 1..n, k = 1..n, t = 5..7, s = 2..4 \text{ e } pcf_{j,t} = pci_{k,s}$

Os motoristas devem estar em um mesmo PC no momento da rendição. Quando j e k não utilizam um mesmo veículo nesta ordem ($X_{j,k} = 0$), a restrição é redundante. Caso contrário, k deve começar sua jornada no mesmo PC em que j termina a sua.

$$X_{j,k} + A_{j,t} + A_{k,s} \leq 2 \qquad \text{para } j = 1..n, k = 1..n, t = 5..7, s = 2..4 \text{ e } f_{j,t} > i_{k,s}$$

Dois motoristas só podem utilizar um mesmo ônibus se a jornada de um termina antes que a do outro comece. A restrição é redundante quando os motoristas j e k não utilizam o mesmo veículo nesta ordem ($X_{j,k} = 0$).

$$E_j \geq F_j - I_j - tt \qquad \text{para } j = 1..n$$

Além de limitar o tamanho das jornadas, esta restrição também define E. Não é utilizada a igualdade porque horas extras não podem ser negativas. Como um dos objetivos é minimizar o número de horas extras, E sempre assume o menor valor possível.

$$F_j - I_j \leq tt - tdsc \qquad \text{para } j = 1..n \text{ e } dsc_j = 0$$

Os motoristas que não têm descanso não podem fazer horas extras e devem ter sua jornada normal reduzida pelo tempo mínimo de descanso.

$$I_j \leq A_{j,2} * i_{j,2} + A_{j,3} * i_{j,3} + A_{j,4} * i_{j,4} - (A_{j,2} + A_{j,4}) * tr \qquad \text{para } j = 1..n$$

$$F_j \geq A_{j,5} * f_{j,5} + A_{j,6} * f_{j,6} + A_{j,7} * f_{j,7} + (A_{j,5} + A_{j,7}) * tr \qquad \text{para } j = 1..n$$

As duas últimas restrições definem as variáveis I e F. O início e o fim de cada jornada de motorista dependem dos arcos selecionados. O início das jornadas que começam com rendição ($A_{j,2} + A_{j,4} = 1$) deve ser adiantado a fim de acomodar o tempo necessário para a troca de motoristas, tr. Da mesma forma, para jornadas que terminam com rendição ($A_{j,5} + A_{j,7} = 1$), seu término deve ser prorrogado.

$$dsc_j \geq I_j + mindsc \qquad \text{para } j = 1..n \text{ e } dsc_j \neq 0$$

$$dsc_j \leq I_j + maxdsc \qquad \text{para } j = 1..n \text{ e } dsc_j \neq 0$$

O descanso do motorista deve começar entre $mindsc$ e $maxdsc$ minutos após o início da sua jornada.

$$F_j - I_k \geq tr - M * (1 - X_{j,k} + A_{j,6}) \qquad \text{para } j = 1..n, k = 1..n$$

$$F_j - I_k \leq tr + M * (1 - X_{j,k} + A_{j,6}) \qquad \text{para } j = 1..n, k = 1..n$$

As duas últimas restrições só não são redundantes se $X_{j,k} = 1$ e $A_{j,6} = 0$, ou seja, se os motoristas j e k utilizam o mesmo veículo e j não termina sua jornada na garagem. Conseqüentemente, k não começa sua jornada na garagem. Satisfeitas estas condições, há rendição e a diferença entre o fim da jornada de j e o início da jornada de k deve ser de exatamente tr minutos.

$$i_{k,2} * A_{k,2} + i_{k,4} * A_{k,4} - (f_{j,5} * A_{j,5} + f_{j,7} * A_{j,7}) \leq$$
$$(1 - X_{j,k} + A_{k,3}) * M + dgar_k + dgar2_j \qquad \text{para } j = 1..n, k = 1..n$$

Se dois motoristas j e k utilizam o mesmo veículo e há tempo suficiente entre suas jornadas para o ônibus ir à garagem e voltar, então esse deve ser o procedimento adotado, isto é, não deve haver rendição. A restrição só não é redundante se $X_{j,k} = 1$ e $A_{k,3} = 0$, ou seja, se os motoristas j e k utilizam o mesmo veículo e k não começa sua jornada na garagem. Conseqüentemente, j não termina a sua na garagem. Satisfeitas estas condições, o intervalo de tempo entre o horário de partida da primeira viagem de k e o horário de chegada da última viagem de j deve ser insuficiente para que o ônibus seja levado à garagem e conduzido de volta ao PC, havendo assim, a rendição.

As próximas restrições referem-se à utilização de duplas pegadas. Em linhas que não permitem o uso deste tipo de jornada, as restrições a seguir podem ser excluídas. No entanto, testes mostraram que a presença delas no modelo não implica em perda considerável de desempenho.

$$A_{j,3} \geq DP_{j,k} \qquad \text{para } j = 1..n, k = 1..n$$

$$A_{k,3} \geq DP_{j,k} \qquad \text{para } j = 1..n, k = 1..n$$

$$A_{j,6} \geq DP_{j,k} \qquad \text{para } j = 1..n, k = 1..n$$

$$A_{k,6} \geq DP_{j,k} \qquad \text{para } j = 1..n, k = 1..n$$

Na jornada dupla pegada, o motorista deve pegar o ônibus na garagem em ambos os turnos em que trabalha. Da mesma forma, deve conduzi-lo até a garagem ao final de cada turno.

$A_{j,t}+A_{k,s}+DP_{j,k} \leq 2 \quad$ para $j = 1..n, k = 1..n, t = 5..7, s = 2..4$ e $i_{k,s} < f_{j,t}+intdp$

Uma dupla pegada deve apresentar um intervalo de pelo menos *intdp* minutos entre os turnos. Se j e k não formam dupla pegada, a restrição é redundante. Caso contrário, o fim da jornada de j deve estar no mínimo a *intdp* minutos do início da jornada de k.

$DP_{j,k} = 0 \quad$ para $j = 1..n, k = 1..n$ e $dsc_j \neq 0$

$DP_{j,k} = 0 \quad$ para $j = 1..n, k = 1..n$ e $dsc_k \neq 0$

Sabe-se previamente que uma dupla pegada não pode ser formada a partir de jornadas de motoristas que têm descanso.

$DP_{j,k} \leq X_{j,k} \quad$ para $j = 1..n, k = 1..n$

Se as jornadas dos motoristas j e k formam dupla pegada ($DP_{j,k} = 1$), então são cobertas pelo mesmo ônibus ($X_{j,k} = 1$).

$\sum_{j=1}^{n} \sum_{k=1}^{n} DP_{j,k} \leq ndp/100 * (n - \sum_{j=1}^{n} \sum_{k=1}^{n} DP_{j,k})$

No máximo, $ndp\%$ das jornadas na escala de motoristas podem ser duplas pegadas. Este modelo pode ser usado tanto para escalas que não aceitam duplas pegadas quanto para aquelas que as aceitam. No primeiro caso, basta colocar $ndp = 0$.

$E_j + E_k \leq le + (1 - DP_{j,k}) * le \quad$ para $j = 1..n, k = 1..n$

$E_j + E_k \geq F_j - I_j + F_k - I_k - tt - tt * (1 - DP_{j,k}) \quad$ para $j = 1..n, k = 1..n$

Cada dupla pegada é formada a partir de duas jornadas geradas pelo escalonamento de motoristas. Quando a junção ocorre, deve-se alterar as restrições que definem hora extra e tamanho de jornada, pois agora um motorista é representado pelas variáveis de dois motoristas distintos. Assim, as somas de horas extras e de tempo trabalhado nas duas pegadas não podem ultrapassar os valores máximos permitidos, que são os mesmos da jornada padrão.

$F_j - I_j \leq tdp + (1 - DP_{j,k}) * (tt + le - tdp) \quad$ para $j = 1..n, k = 1..n$

$$F_k - I_k \leq tdp + (1 - DP_{j,k}) * (tt + le - tdp) \qquad \text{para } j = 1..n, k = 1..n$$

Para as jornadas de motorista que formam dupla pegada, estas restrições limitam o tamanho de cada turno. Para as demais, é redundante.

As restrições a seguir também consideram as variáveis DP, porém não podem ser excluídas caso jornadas do tipo dupla pegada não sejam permitidas. Para eliminar as variáveis DP, basta substituí-las por zero.

$$E_j \leq le * \sum_{k=1}^{n}(DP_{j,k} + DP_{k,j}) \qquad \text{para } j = 1..n \text{ e } dsc_j = 0$$

As jornadas de motorista que formam dupla pegada ($\sum_{k=1}^{n}(DP_{j,k} + DP_{k,j}) = 1$) podem apresentar horas extras de no máximo le minutos. Já nas que não tem descanso e não formam dupla pegada, horas extras são proibidas.

$$\begin{aligned} maximize\ & (\sum_{j=1}^{n}\sum_{k=1}^{n} co * X_{j,k}) + (cm * \sum_{j=1}^{n}\sum_{k=1}^{n} DP_{j,k}) - (ce * \sum_{j=1}^{n} E_j) \\ & -\tfrac{1}{M} * \sum_{j=1}^{n}(A_{j,3} + A_{j,6}) \end{aligned}$$

O objetivo é economizar ao máximo os principais recursos: ônibus, motoristas e horas extras. A primeira parcela premia duplas de motoristas que utilizam um mesmo ônibus, procurando diminuir o tamanho da frota. A segunda parcela estimula duplas pegadas, procurando diminuir o número de motoristas. A terceira parcela pede poucas horas extras. A última parcela da função objetivo estimula a utilização de rendições, porém, como seu peso é menor ($\frac{1}{M}$), ela não interfere nos tamanhos da frota e da tripulação e nem na quantidade de horas extras. Seu efeito é colocar uma rendição quando se pode usar tanto uma rendição como um par de viagens de ida e volta à garagem. Se o motorista j começa sua jornada na garagem somente quando $A_{j,3} = 1$ e a finaliza na garagem somente quando $A_{j,6} = 1$, minimizando $A_{j,3} + A_{j,6}$ procura-se colocar mais rendições na solução.

A utilização de duplas pegadas foi testada no escalonamento de ônibus com o objetivo único de se ter uma noção do quanto pode afetar as escalas de ônibus e de motoristas de determinada linha. Por isso, na redistribuição de horários não foram consideradas as jornadas do tipo dupla pegada, conforme se pode notar na próxima seção. O capítulo 5 mostra a diferença nas escalas de ônibus e de motoristas das linhas entre uma solução que usa duplas pegadas e outra que não usa.

4.4 Redistribuição dos Horários

Esta etapa do algoritmo é crucial para se obter boas soluções. Isso porque se a tabela de horários pode ser eficientemente modificada a partir de uma solução válida, de forma a se obter uma melhor distribuição de viagens, ganha-se grande liberdade na modelagem dos demais sub-problemas do PPV. Esta idéia nasceu observando-se as soluções alcançadas por [46] que, apesar de apresentarem escalas de motoristas e de ônibus muito boas, ofereciam distribuições de horários aquém do desejado. Os primeiros testes foram realizados com as soluções obtidas em [46] e mostraram que realmente era possível melhorar muito a tabela de viagens, estimulando então, a abordagem proposta no início do capítulo 4. Na verdade, o módulo redistribuidor de viagens foi o primeiro a ser implementado, vindo a evoluir bastante até sua versão final.

Dois modelos baseados em um mesmo conjunto de restrições foram propostos para redistribuir os horários. Inicialmente, construiu-se um modelo de programação por restrições devido às vantagens já citadas no capítulo 2. Porém, o sistema mostrou não ser eficiente para todas as instâncias do problema. Um modelo de programação linear foi então construído para resolver este sub-problema do PPV. Esse último modelo atingiu melhores desempenho e qualidade de solução. A programação linear revelou-se superior à programação por restrições em domínio finito para resolver a redistribuição de horários devido, principalmente, às seguintes características do modelo:

- Utiliza muitas variáveis com domínios grandes e que não precisam ser inteiras.
- Não apresenta simetria.
- As restrições e a função objetivo podem ser facilmente formuladas como expressões lineares.

Antes porém de se fazer a redistribuição de viagens, novas viagens devem ser inseridas na tabela de horários.

4.4.1 Inserção de Viagens

No escalonamento de motoristas, as restrições não garantem a intercalação de viagens partindo ora de um PC, ora do outro, mas deixam espaço suficiente entre duas viagens sucessivas de um mesmo PC para que uma viagem possa ser inserida entre elas partindo do PC oposto, e usando o mesmo veículo. Esta viagem é inserida nesta etapa do algoritmo, procurando uniformizar o tempo ocioso entre ela e a viagem anterior, e entre ela e a viagem posterior (veja figura 4.5.a). Se houver espaço, desde que mantenham a intercalação de viagens nos PCs, mais viagens podem ser inseridas (veja figura 4.5.b). Mesmo

que as viagens já estejam intercaladas, um ou mais pares de viagens são inseridos caso haja intervalo de tempo suficiente entre o término de uma viagem e o início da viagem seguinte (veja figura 4.5.c). Isso ajuda a evitar que muitos veículos fiquem estacionados simultaneamente em um mesmo PC, e também que haja tempo ocioso desnecessário. As *horas ociosas*[2] são os períodos em que o motorista espera no PC para efetuar a próxima viagem.

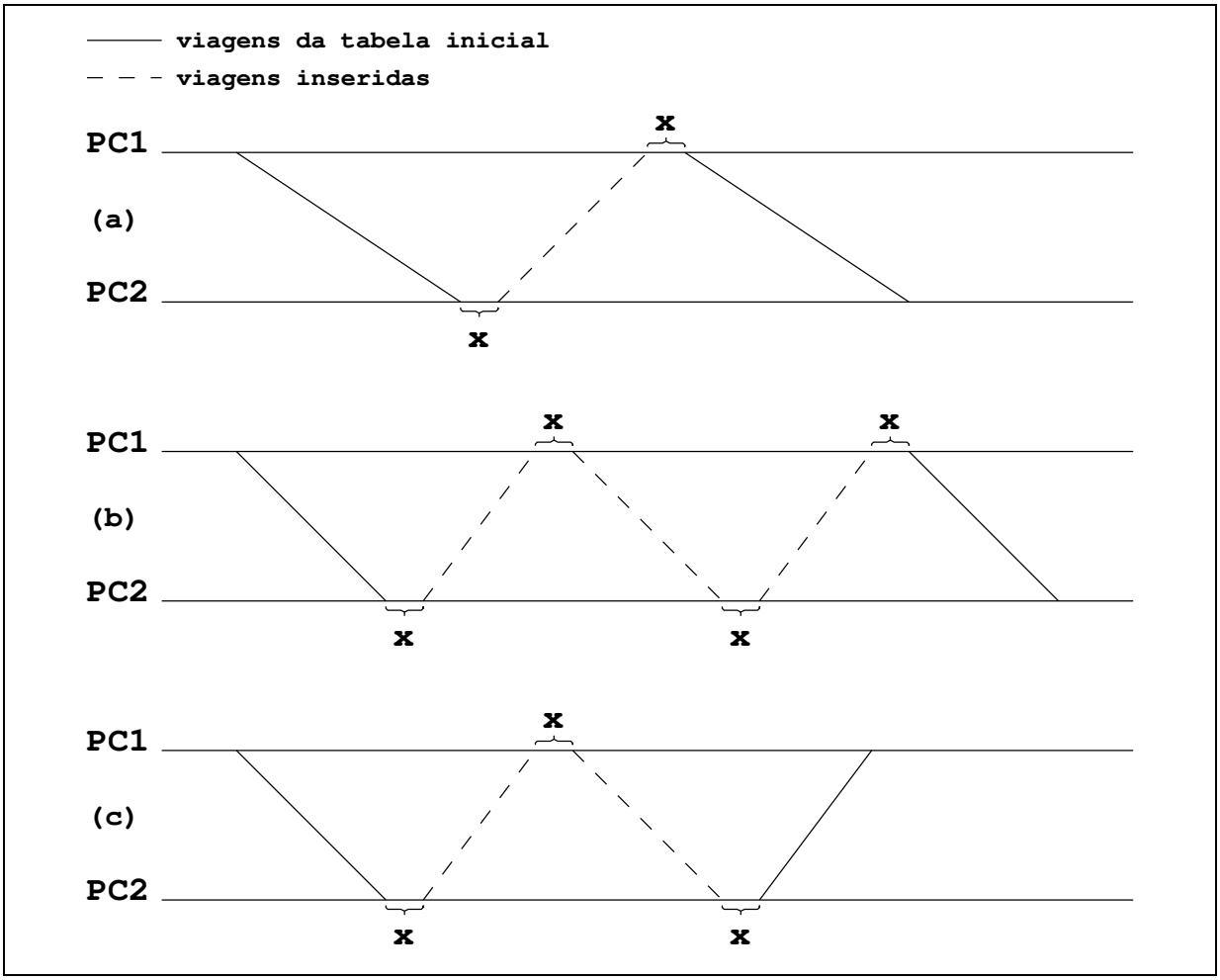


Figura 4.5: Inserção de viagens

No escalonamento de ônibus, outras viagens entre PCs podem ser inseridas. Para cada arco 4 ou 7 (veja a figura 4.4) selecionado, uma viagem é inserida na tabela inicial, pois estes tipos de arco representam a necessidade de uma nova viagem para que os motoristas envolvidos em uma rendição estejam realmente em um mesmo PC.

[2]Diferentemente, em [46], as horas ociosas são aquelas em que o funcionário não trabalha quando faz uma jornada inferior à jornada normal, mas que são efetivamente pagas.

Ainda foi testada uma versão do algoritmo em que o modelo de escalonamento de ônibus não usa a restrição que proíbe rendição nos casos em que há tempo suficiente para o ônibus ir à garagem e voltar. Neste caso, a escala de ônibus pode apresentar rendições realizadas bem antes da primeira viagem da jornada seguinte ou bem depois da última viagem da jornada anterior. Pares de viagens então, podem ser inseridos antes ou depois da rendição, conforme mostra a figura 4.6. Nos testes, porém, quando a restrição foi mantida, obtiveram-se melhores resultados.

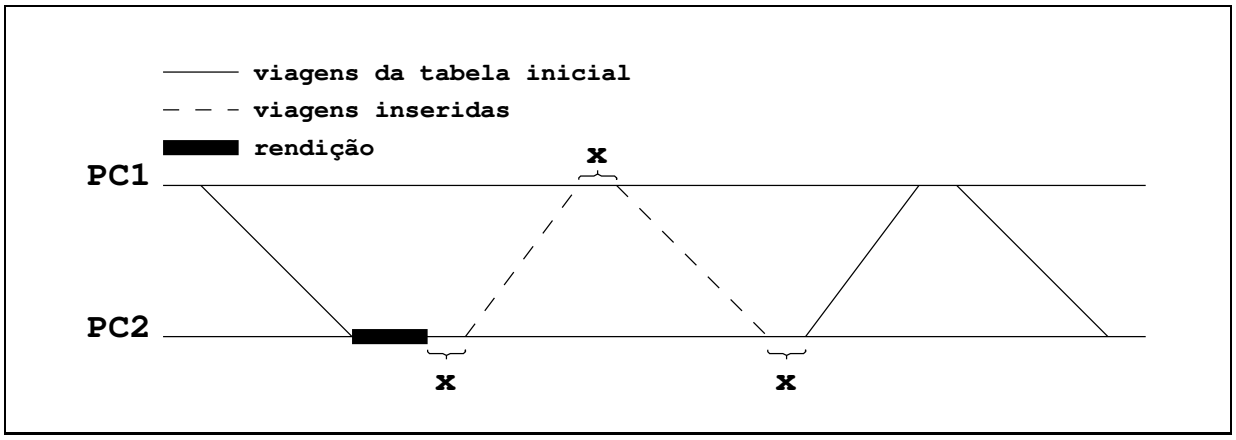


Figura 4.6: Inserção de viagens vizinhas à rendição

A inserção de viagens também pode ser utilizada para atender a novas restrições. Em instâncias que exigem um tempo mínimo de jornada de trabalho para os motoristas, por exemplo, a inserção de viagens poderia fazer com que esta restrição fosse atendida, eliminando esta função dos escalonamentos de ônibus e de motoristas. A inserção também poderia ser usada para forçar a realização de viagens por todos os motoristas nos horários de pico. Assim, inserir viagens nas escalas de ônibus e motoristas é um recurso que ainda pode ser bastante explorado.

4.4.2 Modelo de Programação Linear

A seguir, são explicados os parâmetros, as variáveis e as restrições do modelo. As restrições relativas à função objetivo são abordadas por último.

Parâmetros

n - Quantidade de ônibus necessários para atender à linha.
nv - O ônibus i efetua $nv_{i,k}$ viagens na jornada k.
nm - O ônibus i é utilizado por nm_i motoristas.
d - $d_{i,j,k}$ é a duração da viagem $X_{i,j,k}$. As variáveis X são explicadas logo adiante.

xiahd - A viagem imediatamente anterior ao descanso do motorista da k-ésima jornada no veículo i é determinada pelo índice $xiahd_{i,k}$. Portanto, este parâmetro só existe para as jornadas de motoristas que têm descanso.
tp - $tp_{i,k}$ é o *tempo de posicionamento* do veículo i na jornada k. Refere-se ao tempo necessário para que o veículo fique disponível no PC de onde parte a primeira viagem da jornada k do veículo. Pode ser o tempo de deslocamento do veículo da garagem até o PC ou o tempo de rendição se o veículo é rendido no próprio PC de onde sai a viagem inicial.
tf - $tf_{i,k}$ é o *tempo de finalização* do veículo i na jornada k. Refere-se ao tempo necessário para o motorista liberar o veículo. Pode ser o tempo de deslocamento do ônibus do PC até a garagem ou o tempo de rendição se o ônibus é rendido no próprio PC onde a próxima jornada deve começar.
le - Limite de minutos extras que um motorista pode fazer por dia.
tt - Tempo de trabalho em uma jornada normal.
mindsc - Menor tempo de jornada após o qual se pode iniciar o descanso do motorista.
maxdsc - Maior tempo de jornada antes do qual se pode iniciar o descanso do motorista.
tdsc - Tempo mínimo de descanso.
tr - Tempo de rendição.
dsc - $dsc_{i,k} = 1$ sse o descanso é concedido ao motorista da jornada k do veículo i.
r - $r_{i,k} = 1$ sse a jornada k do veículo i precede uma rendição.

Variáveis

X - $X_{i,j,k}$ é o momento em que parte a j-ésima viagem do veículo i na jornada k.
I - $I_{i,k}$ é o instante de início da k-ésima jornada do veículo i.
F - $F_{i,k}$ é o instante em que a k-ésima jornada do veículo i acaba.
D - $D_{i,k}$ é o momento em que o motorista da k-ésima jornada do veículo i começa seu descanso.
E - $E_{i,k}$ é a quantidade de minutos extras necessária para completar a jornada k do veículo i. Assume valores entre 0 e le.

As variáveis X correspondem aos horários de partidas das viagens e admitem valores que representam minutos do dia. Assim, se uma delas é instanciada com o valor 320, a viagem relacionada a ela parte às 5h20min (320 div 60=5h;320 mod 60=20min). As variáveis I, F e D também correspondem a minutos do dia.

Uma vez que os dados são fornecidos em função da faixa horária, é natural que as viagens também sejam divididas de acordo com o mesmo parâmetro, o que permite uma representação mais simples do problema e a redução do espaço de busca da solução, sem descartar as soluções que são mais prováveis de serem ótimas. Portanto, o domínio de cada variável X é determinado pela sua faixa horária. Por exemplo, qualquer variável que

representa uma viagem entre as 6:01 e as 7h tem um domínio inicial que vai do minuto 361 ao 420.

Restrições

Ao definir as restrições deste sub-problema deve-se levar em consideração que todas as restrições do PPV devem ser respeitadas. No entanto, o modelo para a redistribuição de viagens é gerado a partir de uma solução em que todas as restrições são satisfeitas, exceto a de quase uniformidade na distribuição dos horários de saída das viagens. Se o modelo é construído seguindo certas regras, ele já atende a algumas das restrições implicitamente, as quais são listadas a seguir:

- A demanda deve ser atendida dentro do padrão de qualidade.

  A demanda é fornecida em função da faixa horária. Então, se a redistribuição de viagens mantém a quantidade de viagens por faixa horária em cada PC, a demanda é adequadamente atendida pela nova solução. Como o domínio de cada variável X corresponde à sua faixa horária, esta restrição nunca é violada.

- O ônibus deve sair da garagem quando começa suas atividades e voltar para ela quando não é mais utilizado no dia.

  As viagens que envolvem a garagem não são removidas. O máximo que pode acontecer com elas é ter seu horário de partida alterado. Portanto, a escala de ônibus permanece idêntica à que foi gerada anteriormente.

- Limite de horas extras.

  As variáveis relativas às horas extras são limitadas na declaração do domínio por 0 e le.

- Boa distribuição de viagens.

  Alcançar a melhor distribuição de viagens possível é o objetivo do modelo.

As demais restrições devem ser especificadas no modelo:

1. $X_{i,j+1,k} \geq X_{i,j,k} + d_{i,j,k}$ para $i = 1..n, k = 1..nm_i, j = 1..nv_{i,k} - 1$

   Garantia de que um mesmo ônibus não seja utilizado em duas viagens que ocorrem simultaneamente.

2. $I_{i,k} \leq X_{i,1,k} - tp_{i,k}$ para $i = 1..n, k = 1..nm_i$

3. $F_{i,k} \geq X_{i,nv_{i,k},k} + d_{i,nv_{i,k},k} + tf_{i,k}$ para $i = 1..n, k = 1..nm_i$

   Definições de início e fim de jornada de motorista.

4. $D_{i,k} - I_{i,k} \leq maxdsc$ para $i = 1..n, k = 1..nm_i$ e $dsc_{i,k} = 1$

5. $D_{i,k} - I_{i,k} \geq mindsc$ para $i = 1..n, k = 1..nm_i$ e $dsc_{i,k} = 1$

   O descanso do motorista, quando concedido, se dá entre os minutos $mindsc$ e $maxdsc$ após o início de sua jornada.

6. $D_{i,k} - X_{i,xiahd_{i,k},k} \geq d_{i,xiahd_{i,k},k}$ para $i = 1..n, k = 1..nm_i$ e $dsc_{i,k} = 1$

7. $X_{i,xiahd_{i,k}+1,k} - D_{i,k} \geq tdsc$ para $i = 1..n, k = 1..nm_i$ e $dsc_{i,k} = 1$

   As duas últimas restrições determinam as viagens antecessora e sucessora do descanso. A duração mínima deste, conforme a última restrição, é de $tdsc$ minutos.

8. $F_{i,k} - I_{i,k} \leq tt + E_{i,k}$ para $i = 1..n, k = 1..nm_i$ e $dsc_{i,k} = 1$

9. $F_{i,k} - I_{i,k} \leq tt - tdsc$ para $i = 1..n, k = 1..nm_i$ e $dsc_{i,k} = 0$

10. $E_{i,k} = 0$ para $i = 1..n, k = 1..nm_i$ e $dsc_{i,k} = 0$

    A jornada normal de trabalho é de tt minutos (8). Caso o descanso não seja concedido, ela é encurtada (9) e horas extras não são permitidas (10).

11. $F_{i,k} - I_{i,k+1} = tr$ para $i = 1..n, k = 1..nm_i - 1$ e $r_{i,k} = 1$

    Os motoristas envolvidos em uma rendição devem ter um tempo de tr minutos para fazerem a troca no ônibus.

Restrições relativas à função objetivo

O objetivo do modelo é alcançar a melhor distribuição de viagens possível, o que significa aproximar ao máximo a solução da distribuição ideal, já explicada na seção 3.1. Porém, normalmente, não se encontra uma solução do problema que apresenta tal distribuição.

Uma possível maneira de medir a qualidade de uma solução é calcular seu custo, que pode ser definido como a soma dos desvios de cada viagem em relação ao horário de partida correspondente na distribuição ideal. Quanto menor o custo, melhor a solução. No entanto, como as viagens não influenciam as viagens adjacentes, soluções que na prática são piores podem apresentar custos menores, conforme o exemplo ilustrado na figura 4.7. Provavelmente, um planejador de linhas de ônibus iria preferir a distribuição

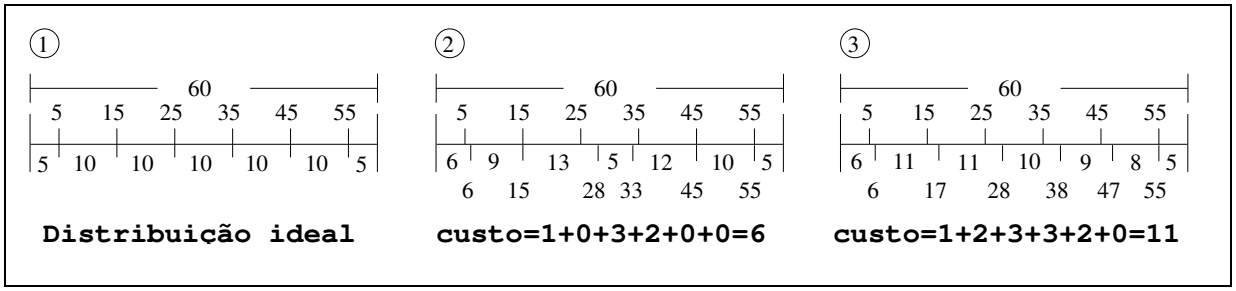


Figura 4.7: Otimização da distribuição das viagens utilizando pontos fixos

3 à distribuição 2, apesar desta apresentar custo menor do que aquela. Na distribuição 3, os espaçamentos entre viagens são mais homogêneos do que na distribuição 2.

Outra maneira seria contabilizar a quantidade de viagens que incidem num ponto diferente do seu horário ideal. Mas, assim como no método anterior, as viagens não influenciariam suas vizinhas.

O problema, portanto, está em adotar pontos fixos como ideais, já que estes permanecem os mesmos ainda que as viagens adjacentes assumam valores distantes de seus ideais. Passou-se a considerar, então, o uso de valores ideais para o espaçamento entre viagens consecutivas, calculando-se o custo não mais para cada viagem, mas para cada par de viagens adjacentes, o que implica em maior influência da viagem em sua vizinhança. No modelo proposto, a função objetivo compara os espaçamentos entre viagens consecutivas a seus adjacentes. Assim, para n viagens em um PC, há $n-1$ variáveis relativas aos espaçamentos entre viagens e $n-2$ relativas às diferenças entre os espaçamentos (veja figura 4.8). O objetivo do modelo é minimizar as diferenças entre os espaçamentos consecutivos, as quais devem assumir valores absolutos para que o custo da solução seja adequadamente calculado. O valor absoluto de uma variável é alcançado conforme mostra o modelo no algoritmo 4.7.

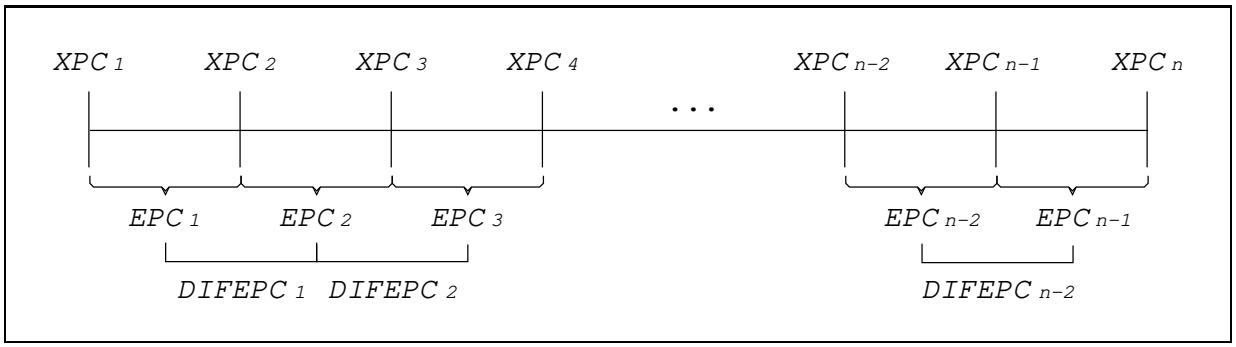


Figura 4.8: Variáveis envolvidas na função objetivo do modelo PL

Outros parâmetros e variáveis são definidos para formular as restrições relativas à função objetivo.

var I;
var $I^+ \geq 0$;
var $I^- \geq 0$;
minimize $I^+ + I^-$;
subject to $I = I^+ - I^-$;
I satisfaz um conjunto de restrições $\mathcal{R}(I)$

Algoritmo 4.7: Valor absoluto de uma variável

Parâmetros

fator - Observando a distribuição ideal ilustrada na figura 3.2, percebe-se que o espaçamento ideal não é o mesmo para todos os pares de viagens consecutivas, pois depende das faixas horárias onde se localizam as viagens. Caso isso fosse ignorado, a distribuição ideal, que deve ter um custo total de zero, teria custo 10, correspondente à soma das diferenças entre os espaçamentos consecutivos [4*(10-10)+(11-10)+(12-11)+3*(12-12)+(16-12)+(20-16)+(20-20)=4*0+1+1+3*0+4+4+0].

Os parâmetros $fator$ tornam possível a comparação dos espaçamentos que envolvem viagens de faixas horárias distintas. O fator de uma faixa horária corresponde ao número que, multiplicado pela espera ideal da faixa, resulta num parâmetro comum a todas as faixas. Portanto, $fator_{fh,p} = k/espera\_ideal_{fh,p}$, em que $espera\_ideal_{fh,p}$ é determinada pela divisão do intervalo da faixa horária fh pela quantidade de viagens partindo de p em fh. Se, por exemplo, $k = 60$, $fator_{fh,p}$ das faixas horárias ilustradas na figura 3.2 são respectivamente 6, 5 e 3.

npc - Há npc_p viagens que partem do PC p com destino a outro PC.

fh - $fh_{i,p}$ é a faixa horária em que está a i-ésima viagem que parte do PC p.

iFH - A faixa horária f começa no minuto iFH_f.

maxdifepcrel - Diferença relativa máxima entre variáveis EPC consecutivas. Sejam x e y um par qualquer de espaçamentos adjacentes, então $x \leq y + maxdifepcrel * y$ e $x \geq y - maxdifepcrel * y$.

maxdifepcabs - Diferença absoluta máxima entre variáveis EPC consecutivas. Ou melhor, $maxdifepcabs$ é o valor máximo que uma variável $DIFEPC$ pode assumir.

Variáveis

XPC - $XPC_{i,p}$ é a i-ésima viagem entre PCs que parte de p.

EPC - É necessário considerar o intervalo entre duas viagens consecutivas. Como deseja-

se comparar as variáveis EPC entre si, elas são definidas pelos intervalos entre as partidas das viagens, multiplicados por $fator_{f,pc}$. EPC só assume valores maiores ou iguais a 1.
DIFEPC - $DIFEPC_{i,p}$ é a diferença entre as esperas $EPC_{i+1,p}$ e $EPC_{i,p}$.
DIFEPC$^+$ e DIFEPC$^-$ - São respectivamente as porções positiva e negativa de $DIFEPC$, e têm a mesma função que as variáveis I^+ e I^- do modelo em 4.7. Assumem valores não negativos.

Como todas as variáveis são contínuas, depois que se tem uma solução do modelo, os valores das variáveis que representam instantes do dia são ajustados para valores inteiros. Desta forma, ganha-se muito em eficiência pois o modelo não utiliza variáveis inteiras, e não se perde em qualidade porque os instantes sofrem alterações na casa dos segundos, desprezíveis para o PPV.

Restrições

$EPC_{i,p} = (XPC_{i+1,p} - XPC_{i,p}) * fator_{f,p}$
para $p = 1..2, i = 1..npc_p - 1$ e $fh_i = fh_{i+1} = f$
Definição dos espaçamentos entre viagens que partem na mesma faixa horária.

$EPC_{i,p} = (XPC_{i+1,p} - iFH_{f+1}) * fator_{f+1,p} + (iFH_{f+1} - XPC_{i,p}) * fator_{f,p}$
para $p = 1..2, i = 1..npc_p - 1$ e $fh_{i+1} = fh_i + 1 = f + 1$
São usados dois fatores para espaçamentos entre viagens que estão em faixas horárias distintas, cada qual multiplicando a porção do intervalo que está na faixa correspondente. Note que se os fatores são iguais, esta restrição equivale à anterior.

$DIFEPC_{i,p} = EPC_{i+1,p} - EPC_{i,p}$ para $p = 1..2, i = 1..npc_p - 2$
Definição da diferença entre os espaçamentos consecutivos entre viagens. Por esta restrição, $DIFEPC$ pode ser tanto positivo quanto negativo, mas como se deseja obter seus valores absolutos, as variáveis $DIFEPC$ são utilizadas conforme a variável I do modelo em 4.7.

$DIFEPC_{i,p} = DIFEPC^+_{i,p} - DIFEPC^-_{i,p}$ para $p = 1..2, i = 1..npc_p - 2$
Corresponde à restrição do modelo descrito em 4.7.

$EPC_{i+1,p} \leq EPC_{i,p} + maxdifepcrel * EPC_{i,p}$ para $p = 1..2, i = 1..npc_p - 2$
$EPC_{i+1,p} \geq EPC_{i,p} - maxdifepcrel * EPC_{i,p}$ para $p = 1..2, i = 1..npc_p - 2$
$EPC_{i+1,p}$ tem que ser *maxdifepcrel* vezes menor ou maior que $EPC_{i,p}$. Desta forma, há um limite para a diferença relativa entre os espaçamentos consecutivos.

$DIFEPC^{+}_{i,p} \leq maxdifepcabs$ para $p = 1..2, i = 1..npc_p - 2$
$DIFEPC^{-}_{i,p} \leq maxdifepcabs$ para $p = 1..2, i = 1..npc_p - 2$
Estas restrições, em conjunto com a função objetivo, fazem com que as variáveis $DIFEPC$ valham no máximo *maxdifepcabs*.

minimize $\sum_{p=1}^{2} \sum_{i=1}^{npc_p - 2} (DIFEPC^{+}_{i,p} + DIFEPC^{-}_{i,p}) + \sum_{i=1}^{n} \sum_{k=1}^{nm_i} E_{i,k}$
A função objetivo tem dois propósitos: minimizar a diferença entre os espaçamentos consecutivos e a quantidade de horas extras. As somas $(DIFEPC^{+}_{i,p} + DIFEPC^{-}_{i,p})$ são utilizadas para extrair os valores absolutos das variáveis $DIFEPC$ de maneira análoga ao modelo em 4.7. A prova de que $DIFEPC^{+}_{i,p} + DIFEPC^{-}_{i,p} = |DIFEPC_{i,p}|\ \forall (i,p)$ é fornecida a seguir.

A função objetivo tem a forma do modelo exibido em 4.8. Note que as variáveis $E_{i,k}$ estão relacionadas a *DIFEPC*, mas não a $DIFEPC^{+}$ e $DIFEPC^{-}$, já que estas só aparecem nas seguintes restrições:

var I;
var $I^{+}_{j} \geq 0 \quad \forall j$;
var $I^{-}_{j} \geq 0 \quad \forall j$;
minimize $\sum_j (I^{+}_{j} + I^{-}_{j}) + \Theta(I)$;
subject to $I_j = I^{+}_{j} - I^{-}_{j} \quad \forall j$;
I satisfaz um conjunto de restrições $\mathcal{R}(I)$;
I^{+} e I^{-} satisfazem um conjunto de restrições $\mathcal{R}'(I^{+}, I^{-})$;

Algoritmo 4.8: Valores absolutos de uma ou mais variáveis adicionados a uma função

1. $DIFEPC_{i,p} = DIFEPC^{+}_{i,p} - DIFEPC^{-}_{i,p}$ para $p = 1..2, i = 1..npc_p - 2$
2. $DIFEPC^{+}_{i,p} \geq 0$ para $p = 1..2, i = 1..npc_p - 2$
3. $DIFEPC^{-}_{i,p} \geq 0$ para $p = 1..2, i = 1..npc_p - 2$
4. $DIFEPC^{+}_{i,p} \leq maxdifepcabs$ para $p = 1..2, i = 1..npc_p - 2$
5. $DIFEPC^{-}_{i,p} \leq maxdifepcabs$ para $p = 1..2, i = 1..npc_p - 2$

A 1ª, a 2ª e a 3ª restrições correspondem respectivamente às restrições $I_j = I^{+}_{j} - I^{-}_{j}$ $\forall j$, $I^{+}_{j} \geq 0\ \forall j$ e $I^{-}_{j} \geq 0\ \forall j$ do modelo exibido em 4.8. As duas últimas fazem parte do conjunto $\mathcal{R}'(I^{+}, I^{-})$, que envolve as restrições da forma $X \leq a$, onde X é $DIFEPC^{+}$

ou $DIFEPC^-$, e a uma constante não negativa. Portanto, podemos assumir que $\Theta(I)$ depende dos valores de I_j, i.e., da diferença $I_j^+ - I_j^-$, mas não depende dos valores isolados de I_j^+ e de I_j^-.

Nessas condições, na solução ótima ainda vale $\min\{I_j^+, I_j^-\}= 0$ e $I_j^+ + I_j^- = |I_j| \quad \forall j$. Caso contrário, em uma solução ótima, teríamos $\min\{I_j^+, I_j^-\}= \delta > 0$, para algum j. Então $I_j^+ \geq \delta$, $I_j^- \geq \delta$ e poderíamos tomar $\bar{I}_j^+ = I_j^+ - \delta \geq 0$, $\bar{I}_j^- = I_j^- - \delta \geq 0$. Para $k \neq j$, toma-se $\bar{I}_k^+ = I_k^+$ e $\bar{I}_k^- = I_k^-$. Assim, $\sum_{j=1}^n (\bar{I}_j^+ + \bar{I}_j^-) = \sum_{j=1}^n (I_j^+ + I_j^-) - 2\delta$. Mas $\bar{I}_j = \bar{I}_j^+ - \bar{I}_j^- = I_j^+ - \delta - I_j^- + \delta = I_j^+ - I_j^- = I_j$. Como $\Theta(I)$ não depende dos valores de I_j^+ e I_j^-, esta parcela da função objetivo não seria alterada. Então, $[\sum_{j=1}^n (\bar{I}_j^+ + \bar{I}_j^-) + \Theta(I)] - [\sum_{j=1}^n (I_j^+ + I_j^-) + \Theta(I)] = -2\delta < 0$, contrariando assim a otimalidade da solução inicial.

```
redistribui:-
    busca1(0), // A busca começa com MaxRel=0
     // busca1 fixa o valor mínimo de maxdifepcrel para o qual existe solução
    busca2(0). // A busca começa com MaxAbs=0

busca1(MaxRel):- // MaxRel corresponde ao parâmetro maxdifepcrel
    modelo_PL1(MaxRel), !. // procura solução com maxdifepcrel=MaxRel
busca1(MaxRel):-
    MaxRelInc is MaxRel+0.01, // incremento de 1%
    busca1(MaxRelInc).

busca2(MaxAbs):- // MaxAbs corresponde ao parâmetro maxdifepcabs
    modelo_PL2(MaxAbs), !. // procura solução com maxdifepcabs=MaxAbs
busca2(MaxAbs):-
    MaxAbsInc is MaxAbs+1,
    busca2(MaxAbsInc).
```

Algoritmo 4.9: Redistribuição de horários utilizando programação linear

O algoritmo 4.9, que faz uso do modelo de programação linear, divide-se em duas etapas. Primeiramente, descobre qual é o valor mínimo de *maxdifepcrel* para o qual existe uma solução, e depois o valor mínimo de *maxdifepcabs*. Já ao final desta segunda etapa, tem-se a melhor solução entre as que usam os menores valores de *maxdifepcrel* e *maxdifepcabs*. É importante salientar que a única diferença entre modelo_PL1 e modelo_PL2 é que este apresenta as restrições que limitam o valor absoluto de $DIFEPC$ enquanto aquele não, já que, durante a primeira etapa, procura-se um valor adequado para *maxdifepcrel* sem considerar *maxdifepcabs*.

Devido à definição das variáveis EPC, a ordem em que as viagens partem de cada PC é mantida inalterada. Por causa desta propriedade, soluções possivelmente melhores

são desconsideradas. Futuramente, novos modelos podem ser formulados para evitar esta situação.

Capítulo 5

Resultados Computacionais

Este capítulo apresenta os resultados computacionais alcançados pelas implementações discutidas nos capítulos anteriores. Para efeito de comparação, esses resultados são confrontados com aqueles obtidos em [46], e também com as escalas reais geradas por empresas. Em seguida, é conduzida uma análise das soluções aqui obtidas nos seus aspectos mais importantes.

5.1 Instâncias

Todas as linhas de ônibus testadas operam na Grande São Paulo e apresentam características bem distintas entre si. Foram obtidos os dados reais de sete instâncias. Porém, tem-se a solução completa usada pelas empresas apenas para as instâncias 3 e 7. Somente para essas duas instâncias foi possível analisar detalhadamente as soluções geradas pelas empresas. A linha 4 é a única circular e as demais utilizam dois PCs.

Diferentes valores para os parâmetros de execução do escalonamento de motoristas foram testados a fim de se encontrar o melhor equilíbrio entre qualidade da solução e desempenho do sistema. Os parâmetros são:

- Tempo máximo, em segundos, para a resolução de cada modelo de programação inteira.
- Quantidade máxima de conjuntos de viagens iniciais distintos, para os quais se gera modelos de programação inteira, em cada instanciação parcial.

Cada instância é identificada por um inteiro positivo, e cada *variação de instância* é determinada pela instância e pelos parâmetros de execução. Então, se em determinada variação da instância 3 utiliza-se, respectivamente, os valores 400 e 10 para os parâmetros de tempo e quantidade de conjuntos, essa instância é representada por 3^{400}_{10}.

Algumas vezes, as soluções alcançadas pelo sistema desenvolvido em [46] não atenderam à demanda. Para que uma comparação mais justa pudesse ser feita em relação a este critério, foram consideradas variações de instâncias em que a demanda de cada faixa horária corresponde ao mínimo entre a demanda real e a que é atendida pela solução gerada através daquele sistema. Essas instâncias modificadas são identificadas por um asterisco nas tabelas. Finalmente, nas variações em que são permitidas duplas pegadas, o sufixo DP é acrescentado à sua designação.

5.2 Escalonamentos de motoristas e de ônibus

O primeiro passo do algoritmo, ou seja, a montagem da tabela inicial de horários, executa rapidamente uma vez que simplesmente constrói a distribuição ideal das viagens. Enquanto o escalonamento de motoristas minimiza a quantidade de motoristas e de horas extras, o escalonamento de veículos minimiza a frota de ônibus e também a quantidade de horas extras, além de incentivar a formação de duplas pegadas. As horas extras resultantes ao final destas duas etapas não são definitivas, já que ainda há a redistribuição de horários. Mesmo assim, elas são exibidas nas tabelas de 5.1 à 5.4, pois já são um primeiro indicativo de quantas horas extras serão necessárias no final da execução do algoritmo. A quantidade de motoristas e de ônibus, entretanto, não se altera após a redistribuição de horários.

As tabelas 5.1, 5.2 e 5.4 exibem os resultados alcançados sem a utilização de duplas pegadas, já que essas não são permitidas nestas linhas. Nessas tabelas, a legenda das colunas é a seguinte:
QO - Quantidade de ônibus utilizados.
QM - Quantidade de motoristas utilizados.
HE - Horas extras necessárias, indicadas na forma *hh:mm*, onde *hh* indica horas e *mm* minutos.
DP - Quantidade de jornadas do tipo dupla pegada. Em todas as variações de instâncias das tabelas 5.1, 5.2 e 5.4, $DP = 0$.

Comparando-se os dados na tabela 5.3 com os valores nas tabelas 5.1 e 5.2, é possível discernir o ganho que se obteria caso o recurso de dupla pegada pudesse ser utilizado nas várias linhas.

A solução encontrada na etapa de escalonamento de ônibus pode não ser ótima mas, geralmente, é bastante adequada. O tempo máximo permitido para se resolver cada modelo de PLI é indicado por parâmetros do sistema, um para o escalonamento de motoristas e outro para o escalonamento de ônibus. No escalonamento de motoristas, vários modelos PLI são resolvidos. Portanto, o valor do parâmetro não pode ser muito alto. No escalonamento de ônibus, o tempo máximo pode ser maior, já que apenas um modelo é resolvido.

Linha	QO	QM	HE
1_{10}^{400}	26	42	40:35
2_{10}^{400}	19	33	23:26
3_{10}^{400}	13	24	03:25
4_{10}^{400}	21	38	00:20
5_{10}^{400}	11	21	05:31
6_{10}^{400}	8	14	13:14
7_{10}^{400}	23	42	04:08

Linha	QO	QM	HE
1_{3}^{400}	26	42	40:26
2_{3}^{400}	20	33	23:16
3_{3}^{400}	14	25	03:28
4_{3}^{400}	20	36	05:14
5_{3}^{400}	12	22	10:19
6_{3}^{400}	8	14	13:14
7_{3}^{400}	23	40	04:20

Tabela 5.1: Escalonamento de motoristas e de ônibus para as variações I_{10}^{400} e I_{3}^{400}

Linha	QO	QM	HE
1_{10}^{400*}	22	38	38:38
2_{10}^{400*}	17	31	19:10
3_{10}^{400*}	11	22	04:40
4_{10}^{400*}	18	32	03:08
5_{10}^{400*}	9	17	03:23
6_{10}^{400*}	9	17	10:04
7_{10}^{400*} X	23	45	04:50

Linha	QO	QM	HE
1_{3}^{400*}	23	38	43:35
2_{3}^{400*}	17	31	21:22
3_{3}^{400*}	12	23	03:31
4_{3}^{400*}	18	32	02:38
5_{3}^{400*}	9	17	04:26
6_{3}^{400*}	9	17	10:04
7_{3}^{400*} X	23	39	02:53

Tabela 5.2: Escalonamento de motoristas e de ônibus para as variações $I_{10}^{400}*$ e $I_{3}^{400}*$

Linha	QO	QM	HE	DP
1_{10}^{400}DP	26	40	43:21	2
2_{10}^{400}DP	19	32	24:41	1
3_{10}^{400}DP	13	23	03:33	1
4_{10}^{400}DP	21	38	00:20	0
5_{10}^{400}DP	11	21	05:31	0
6_{10}^{400}DP	8	14	13:14	0
7_{10}^{400}DP	23	41	05:12	1

Linha	QO	QM	HE	DP
1_{10}^{400*}DP	22	37	41:38	1
2_{10}^{400*}DP	17	31	19:22	0
3_{10}^{400*}DP	11	21	04:40	1
4_{10}^{400*}DP	18	32	03:08	0
5_{10}^{400*}DP	9	17	03:23	0
6_{10}^{400*}DP	9	17	10:04	0
7_{10}^{400*}DP X	23	45	04:50	0

Tabela 5.3: Escalonamento de motoristas e de ônibus para as variações I_{10}^{400}DP e $I_{10}^{400}*$DP

Nos testes realizados, os modelos PLI do escalonamento de motoristas tiveram, como pode ser visto na própria representação das variações de instância, no máximo 200 ou 400 segundos para ser resolvidos. Cada modelo PLI de escalonamento de ônibus despendeu no máximo uma hora. No entanto, esse tempo não foi suficiente para achar sequer uma solução nas variações 7_{10}^{400*}, 7_{3}^{400*}, 7_{10}^{400*}DP, 1_{10}^{200} e 7_{10}^{200}, que são identificadas por um X nas tabelas. A fim de se obter uma solução para estas variações de instâncias em, no máximo,

Linha	QO	QM	HE
1_{10}^{200} X	26	42	37:54
2_{10}^{200}	19	32	34:24
3_{10}^{200}	13	24	03:25
4_{10}^{200}	21	39	05:20
5_{10}^{200}	12	22	10:19
6_{10}^{200}	8	14	13:14
7_{10}^{200} X	24	47	03:36

Tabela 5.4: Escalonamento de motoristas e de ônibus para a variação I_{10}^{200}

uma hora, suprimiu-se do modelo de escalonamento de ônibus a seguinte restrição:
(a) $F_j - I_k \leq tr + M * (1 - X_{j,k} + A_{j,6})$ para $j = 1..n, k = 1..n$.
A restrição acima, juntamente com:
(b) $F_j - I_k \geq tr - M * (1 - X_{j,k} + A_{j,6})$ para $j = 1..n, k = 1..n$,
garante que os dois motoristas envolvidos em uma rendição ocupem, ao mesmo tempo, exatamente tr minutos de suas jornadas nessa atividade. Essas restrições já foram citadas na página 51. Abaixo, são relembradas as definições de variáveis e parâmetros utilizados em a e b, e em restrições relacionadas a elas.
i e **f** - Cada arco determinado pela jornada do motorista j e pelo tipo $t \in 1..7$ tem um momento de início $i_{j,t}$ e de fim $f_{j,t}$.
tr - Tempo de rendição.
X - $X_{j,k} = 1$ sse os motoristas j e k utilizam o mesmo veículo, j antes de k.
I - I_j é o instante de início da jornada do motorista j.
F - F_j é o instante final da jornada do motorista j.
A - $A_{j,t} = 1$ sse o arco do tipo $t \in 1..7$ faz parte da jornada do motorista j.
M - Constante utilizada que representa um número positivo suficientemente grande.

A restrição b, isoladamente, não assegura que os motoristas dediquem simultaneamente tr minutos à rendição, pois situações como a representada na figura 5.1 podem ocorrer. Considere uma rendição envolvendo os motoristas j e k, sendo u_j o momento em que termina a última viagem de j, e p_k o instante em que parte a primeira viagem de k. Observe que, na figura 5.1, valem:

- $I_k \leq p_k - tr$
- $F_j \geq u_j + tr$
- $F_j - I_k \geq tr$

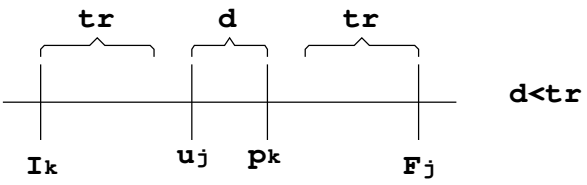


Figura 5.1: Possível cenário se a restrição (a) é removida

Logo, suprimida a restrição a, o cenário ilustrado na figura satisfaz a todas as restrições do problema que consideram o tempo de rendição, já explicadas na página 50, quais sejam:

- $I_j \leq A_{j,2} * i_{j,2} + A_{j,3} * i_{j,3} + A_{j,4} * i_{j,4} - (A_{j,2} + A_{j,4}) * tr$ para $j = 1..n$
- $F_j \geq A_{j,5} * f_{j,5} + A_{j,6} * f_{j,6} + A_{j,7} * f_{j,7} + (A_{j,5} + A_{j,7}) * tr$ para $j = 1..n$
- $F_j - I_k \geq tr - M * (1 - X_{j,k} + A_{j,6})$ para $j = 1..n, k = 1..n$

Portanto, a restrição b, quando não acompanhada de a, garante que os motoristas envolvidos em uma rendição dediquem pelo menos tr minutos a essa atividade, mas não simultaneamente. Eliminando a restrição a, o resolvedor de programação linear inteira encontra, em uma hora, soluções para todas as variações de instância para as quais o modelo completo não conseguiu descobrir solução. É importante observar que eliminar esta restrição pode ser considerado aceitável, uma vez que as próprias empresas a violam em algumas linhas de ônibus.

As tabelas de 5.1 a 5.4 admitem a seguinte análise:

1. I_{10}^{400} *vs.* I_3^{400}: Esperava-se que, em todas as instâncias, I_{10}^{400} apresentasse resultados melhores ou equiparados à variação I_3^{400}, pois considera mais conjuntos de viagens iniciais em cada instanciação parcial. Na maioria dos testes, a variação I_{10}^{400} sobressaiu-se. Porém, as variações 4_3^{400}, 7_3^{400} e 7_3^{400*} apresentaram melhores resultados do que 4_{10}^{400}, 7_{10}^{400} e 7_{10}^{400*}, respectivamente (veja as tabelas 5.1 e 5.2). Se a mesma instanciação parcial aparece na execução das variações I_{10}^{400} e I_3^{400}, certamente a solução para a instanciação parcial da primeira será melhor ou igual à solução da segunda. No entanto, quando as instanciações parciais são diferentes para as duas variações, não há como prever qual delas apresentará melhor solução. Considere duas variações de determinada instância I_{10} e I_3. Suponha que a melhor solução encontrada para a primeira instanciação parcial de I_{10} tenha sido a que utiliza a sétima combinação de viagens iniciais. A solução para a primeira instanciação parcial de I_3 certamente será pior, já que nesta variação considerar-se-á somente até a terceira combinação. A segunda instanciação parcial já não será idêntica para

as duas variações, pois as soluções da primeira instanciação parcial já são diferentes. Portanto, nada certifica que as soluções das instanciações parciais subseqüentes sejam melhores para a variação I_{10} do que para I_3.

2. *I vs. I**: As variações I^* apresentam demandas menores ou iguais às demandas das variações I. Portanto, é esperado que as soluções de I^* usem menos recursos que I. Isso não ocorreu com as variações da instância 6 e com os pares de variações (7_{10}^{400}, 7_{10}^{400*}) e (7_{10}^{400}DP, 7_{10}^{400*}DP). Demandas diferentes dão origem a tabelas de horários diferentes que, por sua vez, exigem escalas de motoristas e de ônibus diferentes. Mesmo que a quantidade de viagens da tabela inicial de uma variação de instância seja menor do que de outra, não significa que menos recursos sejam exigidos para cobri-la, pois as viagens podem estar distribuídas de tal forma que dificultem a reutilização de um mesmo recurso em viagens diferentes.

3. *I vs. I*DP: Obviamente, a formação de duplas pegadas reduz o quadro de motoristas. O efeito colateral é um pequeno aumento no total de horas extras.

4. I^{400} *vs.* I^{200}: Esperava-se que I^{400} expressasse vantagem, ou se equiparasse, a I^{200} em todas as instâncias, uma vez que, na configuração com 400s, os modelos de escalonamento de motoristas dispõem do dobro de tempo para encontrar soluções. A expectativa foi confirmada na maioria das instâncias. Porém, na linha 2, ocorreu uma ligeira melhora na solução de 2_{10}^{200}. Um fator que contribuiu para esse comportamento foi o uso de instanciações parciais distintas, como já discutido no comparativo I_{10}^{400} vs. I_3^{400}.

5.3 Redistribuição de horários

As tabelas 5.5, 5.6 e 5.7 mostram os resultados alcançados com o modelo PL. A legenda de colunas é a seguinte:
DRM - Diferença relativa máxima entre intervalos consecutivos de viagens. Corresponde ao parâmetro *maxdifepcrel* do modelo de programação linear para a redistribuição de horários, explicado na seção 4.4.2. Sejam x e y um par qualquer de espaçamentos adjacentes, então $x \leq y + DRM * y$ e $x \geq y - DRM * y$.
DAM - Diferença absoluta máxima entre intervalos consecutivos de viagens, em minutos. Corresponde ao parâmetro *maxdifepcabs* do modelo de programação linear para a redistribuição de horários, explicado na seção 4.4.2. É o valor máximo que as variáveis *DIFEPC* podem assumir. Elas são definidas na página 62.
DRm - Diferença relativa média entre intervalos consecutivos de viagens.
DAm - Diferença absoluta média entre intervalos consecutivos de viagens.

CD - O custo da distribuição é o valor da função objetivo da redistribuição de horários. Essa informação foi incluída na tabela para permitir uma comparação mais precisa entre as variações de uma mesma instância.

Linha	DRM	DAM	DRm	DAm	CD
1_{10}^{400}	2%	1	0.57%	0.34	2355.75
2_{10}^{400}	26%	27	2.36%	1.64	1653.81
3_{10}^{400}	9%	6	3.11%	1.86	425.458
4_{10}^{400}	0%	0	0.00%	0.00	16.4405
5_{10}^{400}	1%	1	0.25%	0.15	291.629
6_{10}^{400}	7%	5	1.43%	0.91	900.768
7_{10}^{400}	1%	1	0.34%	0.20	265.359

Linha	DRM	DAM	DRm	DAm	CD
1_{3}^{400}	7%	5	2.21%	1.37	2577.66
2_{3}^{400}	4%	2	1.68%	1.01	1555.92
3_{3}^{400}	23%	18	6.73%	4.34	776.538
4_{3}^{400}	0%	0	0.00%	0.00	326.616
5_{3}^{400}	5%	3	0.70%	0.43	651.488
6_{3}^{400}	7%	5	1.43%	0.91	900.768
7_{3}^{400}	37%	34	2.06%	1.54	422.495

Tabela 5.5: Redistribuição de horários nas variações I_{10}^{400} e I_{3}^{400}

Linha	DRM	DAM	DRm	DAm	CD
1_{10}^{400*}	5%	4	2.74%	1.68	2607.63
2_{10}^{400*}	8%	6	2.97%	1.84	1480.29
3_{10}^{400*}	12%	8	4.54%	2.80	632.446
4_{10}^{400*}	1%	1	0.16%	0.09	174.443
5_{10}^{400*}	4%	3	1.08%	0.66	343.351
6_{10}^{400*}	10%	7	2.11%	1.32	849.223
7_{10}^{400*} X	5%	4	1.52%	0.91	369.69

Linha	DRM	DAM	DRm	DAm	CD
1_{3}^{400*}	4%	3	1.79%	1.10	2847.58
2_{3}^{400*}	5%	3	2.03%	1.24	1518.79
3_{3}^{400*}	21%	18	4.68%	3.02	611.296
4_{3}^{400*}	1%	1	0.07%	0.04	155.805
5_{3}^{400*}	3%	2	1.50%	0.91	403.862
6_{3}^{400*}	10%	7	2.11%	1.32	849.232
7_{3}^{400*} X	1%	1	0.23%	0.13	119.911

Tabela 5.6: Redistribuição de horários nas variações I_{10}^{400*} e I_{3}^{400*}

Linha	DRM	DAM	DRm	DAm	CD
1_{10}^{200} X	31%	31	3.14%	2.18	2487.80
2_{10}^{200}	30%	24	6.39%	4.05	2706.89
3_{10}^{200}	9%	6	3.11%	1.86	425.458
4_{10}^{200}	0%	0	0.00%	0.00	352.025
5_{10}^{200}	5%	3	0.70%	0.43	651.488
6_{10}^{200}	7%	5	1.43%	0.91	900.768
7_{10}^{200} X	1%	1	0.53%	0.31	229.852

Tabela 5.7: Redistribuição de horários na variação I_{10}^{200}

Em geral, a variação I_{10}^{400} tem a melhor tabela de horários. Entretanto, todas as variações de instância apresentaram distribuições de horários muito boas, conforme mostram as tabelas 5.5, 5.6 e 5.7. Nota-se que, em comparação com as demais instâncias da variação I_{10}^{400}, por exemplo, a solução para 2_{10}^{400} apresenta valores altos para *DAM* e *DRM*.

No entanto, isto pode ser conseqüência de poucos desvios pontuais na tabela de horários, o que é evidenciado quando analisados os critérios *DAm* e *DRm*, que correspondem às diferenças médias absoluta e relativa. Repare que os valores de *DAm* e *DRm* para 2^{400}_{10} são até menores do que para 3^{400}_{10}, enquanto *DAM* e *DRM* nesta última instância são bem menores do que naquela.

Na prática, uma única viagem pode comprometer uma tabela de horários inteira se estiver muito deslocada em relação às demais. Por isso, valores baixos para *DRM* e *DAM* são importantes. *DRm* e *DAm* também são exibidos para mostrar que quando *DRM* e *DAM* têm valores um pouco acima do esperado, são poucas as viagens que têm horário de partida mal posicionado.

Para se ter uma noção da qualidade na distribuição de horários das soluções geradas, considere uma faixa horária em que haja três viagens partindo nos minutos 10, 20 e 28. Se os espaçamentos entre as viagens são de 10 e 8 minutos, então a diferença entre os espaçamentos é de apenas dois minutos, o que, na prática, é pequena. A diferença relativa entre os espaçamentos é de 25% pois $(10 = 8 + 25\% * 8)$, $(8 = 10 - 20\% * 10)$ e $\max\{25,20\}=25$. Portanto, para esta solução, *DRM* será pelo menos 25%. Note que o maior valor de *DRM* entre as soluções para as variações I^{400}_{10}, por exemplo, é de 26%. Portanto, é notável a qualidade das distribuições de horários nas soluções.

5.4 Tempos de processamento

O sistema implementado processou as instâncias em uma máquina tipo PC equipada com um processador AMD Athlon de 900 MHz e com 768 MB de RAM. As tabelas de 5.8 a 5.12 mostram o tempo de processamento despendido para resolver o escalonamento de motoristas nas variações consideradas. A legenda de colunas nessa tabela é a seguinte:
TEM - Tempo de processamento necessário para o escalonamento de motoristas, indicado na forma *hh:mm:ss:cc*, onde *hh* indica horas, *mm* minutos, *ss* segundos e *cc* indica centésimos de segundo.
QPIEM - Quantidade de modelos de programação inteira resolvidos durante o escalonamento de motoristas. Não são contabilizados os modelos cuja infactibilidade é detectada já na fase de pré-resolução do CPLEX.
PPREM - Porcentagem do tempo de processamento do escalonamento de motoristas dedicado à programação por restrições.
PPIEM - Porcentagem do tempo de processamento do escalonamento de motoristas dedicado à programação inteira.
QM, previamente explicado, é fornecido para dar uma idéia do tamanho da instância.

Algumas características são facilmente identificadas quando se analisa as tabelas referentes aos tempos de processamento do escalonamento de motoristas. Por exemplo,

Linha	QM	TEM	QPIEM	PPREM	PPIEM
1_{10}^{400}	42	08:18:41.59	31	77.86%	22.14%
2_{10}^{400}	33	04:27:27.64	36	32.53%	67.47%
3_{10}^{400}	24	00:23:48.08	27	65.92%	34.08%
4_{10}^{400}	38	04:11:48.15	36	11.09%	88.91%
5_{10}^{400}	21	03:17:02.23	27	30.65%	69.35%
6_{10}^{400}	14	01:04:04.66	10	88.22%	11.78%
7_{10}^{400}	42	05:44:27.01	44	27.37%	72.63%

Tabela 5.8: Tempo de processamento do escalonamento de motoristas na variação I_{10}^{400}

Linha	QM	TEM	QPIEM	PPREM	PPIEM
1_{3}^{400}	42	07:33:45.77	24	86.18%	13.82%
2_{3}^{400}	33	02:31:30.04	19	55.77%	44.23%
3_{3}^{400}	25	00:19:30.54	13	79.70%	20.30%
4_{3}^{400}	36	02:23:32.38	18	18.81%	81.19%
5_{3}^{400}	22	02:13:23.17	16	45.39%	54.61%
6_{3}^{400}	14	01:03:44.65	10	88.54%	11.46%
7_{3}^{400}	40	03:26:43.30	22	43.93%	56.07%

Tabela 5.9: Tempo de processamento do escalonamento de motoristas na variação I_{3}^{400}

Linha	QM	TEM	QPIEM	PPREM	PPIEM
1_{10}^{400*}	38	09:49:31.36	47	61.06%	38.94%
2_{10}^{400*}	31	04:30:42.38	35	28.48%	71.52%
3_{10}^{400*}	22	00:18:51.68	23	76.82%	23.18%
4_{10}^{400*}	32	03:57:53.16	36	9.09%	90.91%
5_{10}^{400*}	17	02:02:46.06	26	29.11%	70.89%
6_{10}^{400*}	17	03:02:31.86	23	29.68%	70.32%
7_{10}^{400*}	45	06:28:02.02	52	22.74%	77.26%

Tabela 5.10: Tempo de processamento do escalonamento de motoristas na variação $I_{10}^{400}*$

a quantidade de modelos de programação inteira *QPIEM* nas variações I_{10} normalmente é maior do que nas variações I_3. Isso era de se esperar por motivos já expostos na comparação $I_{10}^{400} vs. I_3^{400}$ apresentada na seção 5.2.

Note que as proporções *PPREM* e *PPIEM* não são homogêneas. Elas podem variar muito de uma instância para outra e não há uma relação direta entre essas medidas e as demais, *QM*, *TEM* e *QPIEM*. As diferenças de *PPREM* e *PPIEM* entre variações de uma mesma instância já não são gritantes. A instância 6, entretanto, apresenta valores para

Linha	QM	TEM	QPIEM	PPREM	PPIEM
1_3^{400*}	38	07:29:05.05	28	77.38%	22.62%
2_3^{400*}	31	02:42:28.07	19	46.53%	53.47%
3_3^{400*}	23	00:15:55.89	9	89.62%	10.38%
4_3^{400*}	32	01:39:22.54	12	21.14%	78.86%
5_3^{400*}	17	01:10:41.99	12	50.09%	49.91%
6_3^{400*}	17	02:56:24.45	22	30.93%	69.07%
7_3^{400*}	39	03:33:57.57	24	39.67%	60.33%

Tabela 5.11: Tempo de processamento do escalonamento de motoristas na variação I_3^{400*}

Linha	QM	TEM	QPIEM	PPREM	PPIEM
1_{10}^{200}	42	07:24:25.92	33	87.13%	12.87%
2_{10}^{200}	32	03:07:39.97	41	45.62%	54.38%
3_{10}^{200}	24	00:23:44.61	27	65.84%	34.16%
4_{10}^{200}	39	02:34:40.03	39	17.86%	82.14%
5_{10}^{200}	22	02:16:03.38	27	45.03%	54.97%
6_{10}^{200}	14	01:03:37.26	10	88.57%	11.43%
7_{10}^{200}	47	04:14:55.18	55	37.00%	63.00%

Tabela 5.12: Tempo de processamento do escalonamento de motoristas na variação I_{10}^{200}

PPREM e *PPIEM* muito diferentes quando comparadas as variações I e I^*, talvez porque as alterações nas demandas tenham produzido instanciações parciais muito diferentes.

A tendência é que as instâncias em que são necessários mais motoristas levem mais tempo para serem processadas, mas nem sempre isso ocorre. Repare, por exemplo, que na tabela 5.8, as instâncias que exigem mais motoristas, como a 1 e a 7, são as que necessitam de mais tempo de máquina. Por outro lado, a linha 3 usa quase o dobro de motoristas em relação à linha 6, mas é computada em menos da metade do tempo. Uma desproporcionalidade também pode ser observada entre as linhas 3 e 5. Portanto, é difícil estipular um horizonte de tempo para que o sistema retorne uma solução para o escalonamento de motoristas.

Os tempos de processamento para o escalonamento de ônibus e para a redistribuição de horários também são fornecidos, nas tabelas de 5.13 à 5.15, com a seguinte legenda de colunas:

TEO - Tempo de processamento necessário para o escalonamento de ônibus, em segundos. O tempo máximo permitido foi de uma hora. Nas execuções em que o valor de TEO foi inferior a 3600s, encontrou-se a solução ótima.

TRH - Tempo de processamento necessário para a redistribuição de horários, em segundos.

QPL - Quantidade de modelos de programação linear resolvidos durante a redistribuição de horários.
QR - Quantidade de rendições na solução. Este campo foi adicionado às tabelas porque as rendições influem de maneira considerável o tempo de execução do escalonamento de ônibus.
QO e **QM** são fornecidos para se ter uma idéia do tamanho das instâncias.

Linha	QO	QM	QR	TEO	TRH	QPL
1_{10}^{400}	26	42	5	3600	1.17	5
2_{10}^{400}	19	33	4	3600	8.87	55
3_{10}^{400}	13	24	5	3200	1.91	17
4_{10}^{400}	21	38	11	3600	0.09	2
5_{10}^{400}	11	21	0	7.4	0.60	4
6_{10}^{400}	8	14	0	6.7	2.31	14
7_{10}^{400}	23	42	11	3600	0.51	4

Linha	QO	QM	QR	TEO	TRH	QPL
1_{3}^{400}	26	42	8	3600	3.76	14
2_{3}^{400}	20	33	6	3600	1.54	8
3_{3}^{400}	14	25	5	3600	3.32	43
4_{3}^{400}	20	36	12	3600	0.10	2
5_{3}^{400}	12	22	0	1.8	1.77	10
6_{3}^{400}	8	14	0	6.7	2.38	14
7_{3}^{400}	23	40	6	3600	7.41	73

Tabela 5.13: Tempo de processamento para o escalonamento de ônibus e para a redistribuição de horários nas variações I_{10}^{400} e I_{3}^{400}

Linha	QO	QM	QR	TEO	TRH	QPL
1_{10}^{400*}	22	38	10	3600	3.04	11
2_{10}^{400*}	17	31	9	3600	3.59	16
3_{10}^{400*}	11	22	4	170	2.77	22
4_{10}^{400*}	18	32	9	3600	0.24	4
5_{10}^{400*}	9	17	1	0.9	1.19	9
6_{10}^{400*}	9	17	2	33	3.14	19
7_{10}^{400*} X	23	45	15	3600	1.5	11

Linha	QO	QM	QR	TEO	TRH	QPL
1_{3}^{400*}	23	38	5	3600	2.49	9
2_{3}^{400*}	17	31	7	3600	2.13	10
3_{3}^{400*}	12	23	5	3100	3.98	41
4_{3}^{400*}	18	32	10	3600	0.29	4
5_{3}^{400*}	9	17	0	0.53	0.94	7
6_{3}^{400*}	9	17	2	32	3.1	19
7_{3}^{400*} X	23	39	14	3600	0.46	4

Tabela 5.14: Tempo de processamento para o escalonamento de ônibus e para a redistribuição de horários nas variações I_{10}^{400*} e I_{3}^{400*}

No conjunto de instâncias testadas, as linhas que exigem menos motoristas e ônibus são também as que apresentam menos rendições. Estes dois fatores parecem influenciar significativamente no tempo de processamento do escalonamento de ônibus. Repare que, em uma hora, soluções ótimas foram encontradas para todas as variações das instâncias 5 e 6, e para quatro das cinco variações analisadas da instância 3. Estas são justamente as soluções que apresentam menos rendições nas escalas e as que utilizam menos recursos. É possível identificar faixas de valores para os tempos de execução de acordo com a quantidade de rendições. Em linhas que não precisaram de mais que uma rendição,

Linha	QO	QM	QR	TEO	TRH	QPL
1_{10}^{200} X	26	42	13	3600	17.44	64
2_{10}^{200}	19	32	9	3600	12.72	56
3_{10}^{200}	13	24	5	3200	1.84	17
4_{10}^{200}	21	39	10	3600	0.07	2
5_{10}^{200}	12	22	0	1.8	1.72	10
6_{10}^{200}	8	14	0	6.7	2.43	14
7_{10}^{200} X	24	47	15	3600	0.49	4

Tabela 5.15: Tempo de processamento para o escalonamento de ônibus e para a redistribuição de horários na variação I_{10}^{200}

obteve-se solução ótima em menos de 10 segundos. Por volta de 30 segundos foram suficientes para encontrar a melhor solução nas escalas com duas rendições. Mais de 50 minutos foram necessários para se encontrar soluções ótimas em linhas com 5 rendições, nas instâncias em que foi possível encontrar solução ótima. Nenhuma solução ótima foi encontrada para instâncias com mais de 5 rendições. Duas soluções apresentaram 4 rendições. Na variação 3_{10}^{400*}, em 170 segundos, encontrou-se a solução ótima, o que não foi possível na variação 2_{10}^{400} com uma hora de execução. Isso parece ocorrer porque a instância 2 exige uma quantidade maior de motoristas e de ônibus. Daí, nota-se que a quantidade de rendições e de recursos necessários em uma linha de ônibus são decisivos no tempo de processamento para o escalonamento de ônibus.

Não se percebe nenhuma relação aparente entre *TRH* ou *QPL* com *TEO*, *QO*, *QM* ou *QR*. Portanto, o escalonamento de veículos não influi muito na etapa de redistribuição de horários quanto ao tempo de processamento. *QPL* está relacionada a *TRH*, pois quanto maior a quantidade de modelos a ser resolvidos, maior o tempo necessário de computação. Cada modelo de redistribuição de horários, independentemente da instância, é resolvido em um intervalo de tempo na ordem de décimos de segundo.

5.5 Análise das soluções

A seguir, são apresentadas algumas tabelas com dados acerca de algumas soluções geradas pelo método manual das empresas, de soluções obtidas pelo sistema desenvolvido em [46] e de soluções construídas pelo sistema implementado nesse trabalho. Além de atributos já explicados anteriormente, essas tabelas utilizam a seguinte legenda de colunas:
HO - Quantidade de horas ociosas na escala de motoristas. As *horas ociosas* são os períodos em que o motorista espera no PC pela saída da próxima viagem.

HN - Quantidade de horas normais na escala de motoristas. *Horas normais* correspondem ao tempo total da escala, subtraída as horas extras e ociosas. Tanto HO quanto HN são indicados na forma *hh:mm*, em que *hh* refere-se a horas e *mm* a minutos.
QV - Quantidade de viagens entre PCs na linha.
E1 e **E2** - Empilhamento máximo no PC1 e no PC2. Entenda-se por *empilhamento* a quantidade de ônibus parados em determinado PC no mesmo instante, e por *empilhamento máximo* o empilhamento no instante do dia em que há a maior quantidade de ônibus parados no PC.
DNA - Demanda não atendida. Corresponde à quantidade de faixas horárias cuja demanda não foi adequadamente atendida. As soluções geradas pelo sistema implementado nesse trabalho sempre obtêm $DNA = 0$.
TT - Tempo total de resolução dos modelos. Contabiliza somente as etapas em que os modelos de programação por restrições e de programação linear são resolvidos. O tempo real de processamento é um pouco maior, já que ainda são efetuadas operações de geração de arquivos, compilação de programas e conversão de formatos de arquivos. Porém, deve ser observado que este tempo adicional é irrelevante comparado ao que o sistema usa para processar uma instância. O tempo de processamento só é fornecido para as execuções do sistema implementado neste trabalho. TT é indicado na forma *hh:mm:ss:cc*, onde *hh* indica horas, *mm* minutos, *ss* segundos e *cc* indica centésimos de segundo. Um traço (-) em uma célula da tabela indica que a informação correspondente não estava disponível.

Em [46], as soluções foram geradas a partir de uma execução em que se tentou equipará-las às soluções fornecidas pelas empresas, cujos detalhes são exibidos na tabela 5.16. A pedido [45], as instâncias tratadas aqui foram executadas novamente no sistema de [46] procurando utilizar menos recursos. Os dados acerca destas novas soluções são mostrados na tabela 5.17. E a tabela 5.18 expõe os detalhes das soluções produzidas pelo sistema construído durante este trabalho.

Linha	QO	QM	HE	HO	HN	QV	E1	E2	DNA	DRM	DAM	CD
1	28	-	-	-	-	-	-	-	-	-	-	-
2	25	-	-	-	-	-	-	-	-	-	-	-
3	15	30	18:25	42:17	227:15	232	6	3	0	*	123	3242.03
4	21	42	26:48	-	-	177	-	-	0	-	-	-
5	-	-	-	-	-	-	-	-	-	-	-	-
6	-	-	-	-	-	-	-	-	-	-	-	-
7	26	52	25:28	23:54	358:14	235	5	5	0	*	135	3691.53

*Não há como calcular DRM porque a solução apresenta viagens partindo no mesmo minuto, ou seja, apresenta espaçamento entre viagens nulo

Tabela 5.16: Soluções fornecidas pelas empresas

Linha	QO	QM	HE	HO	HN	QV	E1	E2	DNA	DRM	DAM	CD
1	24	47	21:42	26:30	301:47	265	4	3	3	97%	122	4786.78
2	21	42	15:48	26:18	277:05	261	4	3	1	97%	104	5324.75
3	14	28	18:01	22:57	188:21	222	4	3	1	95%	101	4761.36
4	17	33	09:54	20:43	211:41	219		5	5	92%	150	4145.08
5	7	14	12:31	15:03	102:40	173	4	2	14	88%	116	4320.00
6	7	14	13:13	17:35	98:33	206	3	2	4	94%	106	4229.49
7	26	51	02:15	13:54	320:49	194	4	2	4	91%	109	3914.24

Tabela 5.17: Soluções geradas em [45]

A seguir, a partir destas tabelas, faz-se uma análise das soluções sob vários aspectos.

1. *Atendimento às restrições e à demanda*

   Freqüentemente, as soluções construídas nas empresas violam as restrições do PPV, prejudicando, muitas vezes, os motoristas ou mesmo os passageiros. As soluções geradas pelo sistema implementado neste trabalho atendem estritamente a todas as restrições do PPV, exceto aquelas identificadas na subseção 1.2.4. Também atendem integralmente à demanda devido à forma como o problema é resolvido. Algumas soluções produzidas em [45] e algumas das soluções apresentadas em [46] não atendem à demanda adequadamente em todas as faixas horárias, o que também pode ser verificado em soluções construídas pelas empresas. Entretanto, nas soluções fornecidas pelas empresas para a realização deste trabalho, as demandas são efetivamente atendidas.

2. *Utilização de recursos*

   Os recursos mais importantes para se operar uma linha são: ônibus, motoristas e horas extras. A utilização mínima destes recursos é o principal objetivo do sistema implementado. Suas soluções atingiram uma redução de até 24% no tamanho da frota e de até 20% no tamanho da tripulação em relação às soluções geradas nas empresas. A figura 5.2 ilustra essas diferenças. A ausência de uma barra na figura indica indisponibilidade de dados. Em comparação com as soluções obtidas em [45], representadas na tabela 5.17, as soluções geradas pelo sistema levam vantagem, principalmente na quantidade de motoristas utilizados nas maiores instâncias. Em algumas linhas de ônibus, as soluções de [45] utilizam menos recursos, mas, não raro, violam restrições do problema. Em geral, conforme mostra a figura 5.3, as soluções

Linha	QO	QM	HE	HO	HN	QV	E1	E2	DRM	DAM	CD	TT
1_{10}^{400}	26	42	35:34	50:49	281:18	224	8	6	2%	1	2355.75	09:18:42.76
1_{3}^{400}	26	42	32:37	52:55	280:11	218	6	6	7%	5	2577.66	08:33:49.53
1_{10}^{200} X	26	42	31:44	53:46	287:12	225	6	5	31%	31	2487.80	08:24:43.36
2_{10}^{400}	19	33	21:49	38:34	220:52	185	5	5	26%	27	1653.81	05:27:36.51
2_{3}^{400}	20	33	22:03	41:47	223:23	185	6	6	4%	2	1555.92	03:31:31.58
2_{10}^{200}	19	32	31:18	39:01	216:15	195	5	4	30%	24	2706.89	04:07:52.69
3_{10}^{400}	13	24	02:58	27:52	153:59	133	4	3	9%	6	425.458	01:17:09.99
3_{3}^{400}	14	25	02:57	31:10	161:58	138	3	4	23%	18	776.538	01:19:33.86
3_{10}^{200}	13	24	02:58	27:52	153:59	133	4	3	9%	6	425.458	01:17:06.45
4_{10}^{400}	21	38	00:17	34:31	252:04	193		9	0%	0	16.4405	05:11:48.24
4_{3}^{400}	20	36	04:14	29:52	241:38	196		7	0%	0	326.616	03:23:32.48
4_{10}^{200}	21	39	03:12	36:12	254:28	193		8	0%	0	352.025	03:34:40.10
5_{10}^{400}	11	21	03:49	32:30	142:14	186	4	4	1%	1	291.629	03:17:10.23
5_{3}^{400}	12	22	08:05	35:20	145:07	196	3	3	5%	3	651.488	02:13:26.74
5_{10}^{200}	12	22	08:05	35:20	145:07	196	3	3	5%	3	651.488	02:16:06.90
6_{10}^{400}	8	14	12:10	25:16	101:51	185	2	4	7%	5	900.768	01:04:13.67
6_{3}^{400}	8	14	12:10	25:16	101:51	185	2	4	7%	5	900.768	01:03:53.73
6_{10}^{200}	8	14	12:10	25:16	101:51	185	2	4	7%	5	900.768	01:03:46.39
7_{10}^{400}	23	42	03:48	28:57	270:39	143	3	4	1%	1	265.359	06:44:27.52
7_{3}^{400}	23	40	03:26	23:19	265:28	142	3	4	37%	34	422.495	04:26:50.71
7_{10}^{200} X	24	47	02:49	49:58	299:44	146	5	5	1%	1	229.852	05:14:55.67
1_{10}^{400*}	22	38	35:42	48:32	261:14	212	5	6	5%	4	2607.63	10:49:34.40
1_{3}^{400*}	23	38	41:01	52:36	260:28	210	5	6	4%	3	2847.58	08:29:07.54
2_{10}^{400*}	17	31	18:34	35:42	210:33	180	4	5	8%	6	1480.29	05:30:45.97
2_{3}^{400*}	17	31	21:36	35:53	210:46	181	4	5	5%	3	1518.79	03:42:30.20
3_{10}^{400*}	11	22	04:35	20:02	138:57	128	4	4	12%	8	632.446	00:21:44.45
3_{3}^{400*}	12	23	03:45	26:48	148:23	128	4	4	21%	18	611.296	01:07:39.87
4_{10}^{400*}	18	32	02:25	18:47	215:34	181		7	1%	1	174.443	04:57:53.40
4_{3}^{400*}	18	32	02:09	18:09	215:32	182		7	1%	1	155.805	02:39:22.83
5_{10}^{400*}	9	17	03:46	27:10	118:56	157	3	3	4%	3	343.351	02:02:48.15
5_{3}^{400*}	9	17	04:20	29:01	121:21	159	3	3	3%	2	403.862	01:10:43.46
6_{10}^{400*}	9	17	10:05	30:51	111:31	185	3	3	10%	7	849.223	03:03:08.00
6_{3}^{400*}	9	17	10:05	30:51	111:31	185	3	3	10%	7	849.232	02:56:59.55
7_{10}^{400*} X	23	45	03:58	48:33	296:10	143	6	4	5%	4	369.690	07:28:03.52
7_{3}^{400*} X	23	39	01:37	32:58	263:36	135	4	4	1%	1	119.911	04:33:58.03

Tabela 5.18: Soluções geradas pelo sistema implementado

da tabela 5.18 para as variações I^* das maiores instâncias são mais econômicas na utilização de motoristas e ônibus do que as soluções da tabela 5.17, apesar de usarem mais horas extras em alguns casos, como mostra a figura 5.4. E elas atendem à mesma demanda.

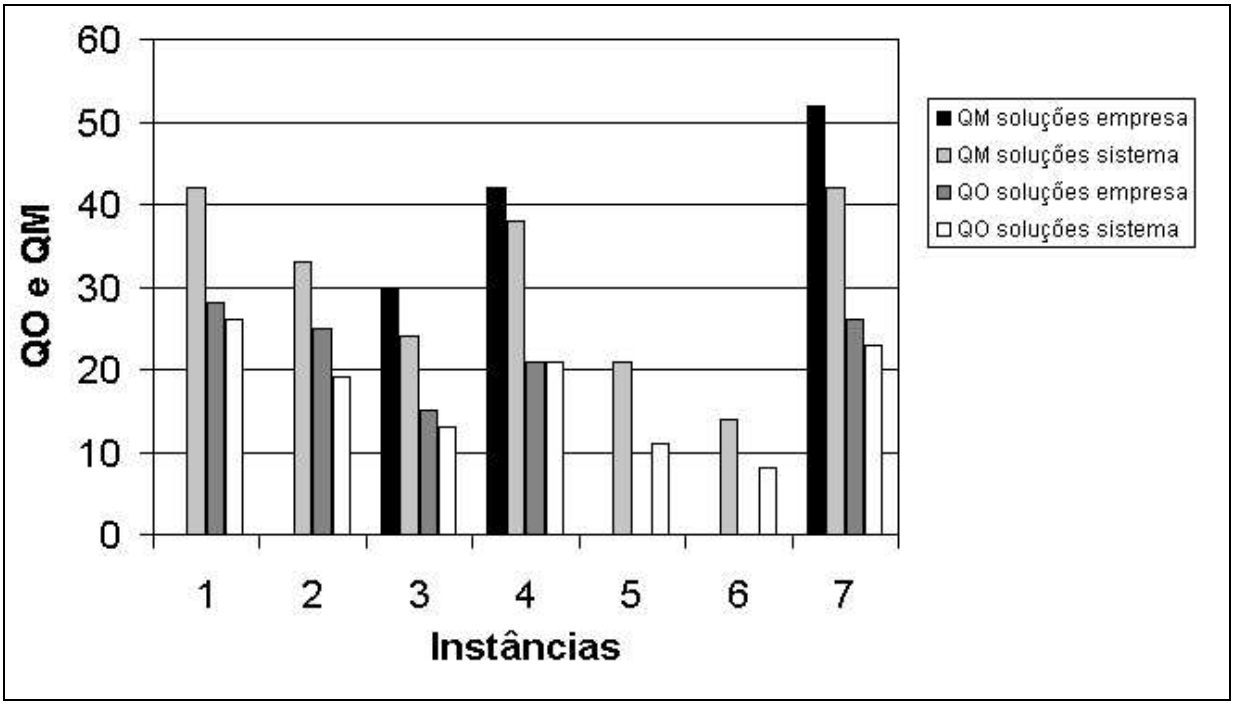


Figura 5.2: Comparativo na utilização de recursos entre as soluções das empresas e I_{10}^{400}

3. *Tabela de horários*

   A distribuição de horários gerada pelo sistema foi muito boa para todas as variações de instâncias. Nesse aspecto, as soluções obtidas pelo sistema construído nesse trabalho apresentaram substancial vantagem em relação às soluções produzidas pelo sistema de [45] e àquelas construídas pelas empresas. Para efeito de comparação, as soluções manuais das empresas e aquelas obtidas em [45] foram submetidas ao módulo de distribuição de horários desenvolvido nesse trabalho. Os resultados são apresentados nas tabelas 5.19 e 5.20. Apesar de apresentarem uma melhora considerável após a redistribuição de horários, elas ainda não atingiram o mesmo nível de qualidade das soluções finais geradas pelo sistema implementado neste trabalho.

Linha	DRM	DAM	DRm	DAm	CD
1	97%	122	20.44%	18.06	4786.78
2	97%	104	23.79%	20.40	5324.75
3	95%	101	27.08%	21.45	4761.36
4	92%	150	17.94%	15.10	4145.08
5	88%	116	28.81%	24.97	4320.00
6	94%	106	24.06%	20.53	4229.49
7	91%	109	25.19%	20.18	3914.24

Linha	DRM	DAM	DRm	DAm	CD
1	43%	31	4.93%	3.29	2136.98
2	42%	22	2.02%	1.28	1272.19
3	16%	11	2.80%	1.78	1435.53
4	18%	15	1.80%	1.24	789.374
5	21%	20	3.78%	2.53	437.576
6	22%	14	3.26%	2.02	1163.89
7	55%	70	4.95%	3.51	799.915

Tabela 5.19: Distribuição de horários nas soluções de [45] antes e depois da redistribuição

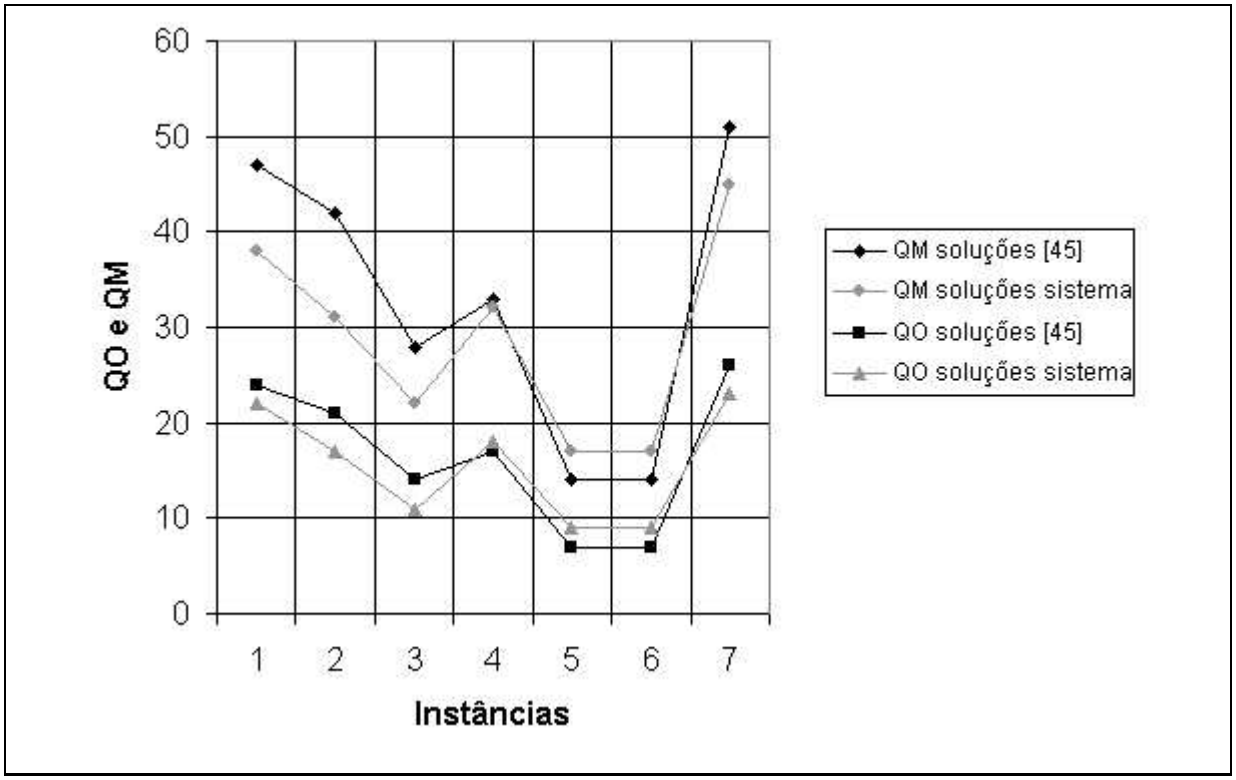


Figura 5.3: Comparativo na utilização de recursos entre as soluções de [45] e I_{10}^{400*}

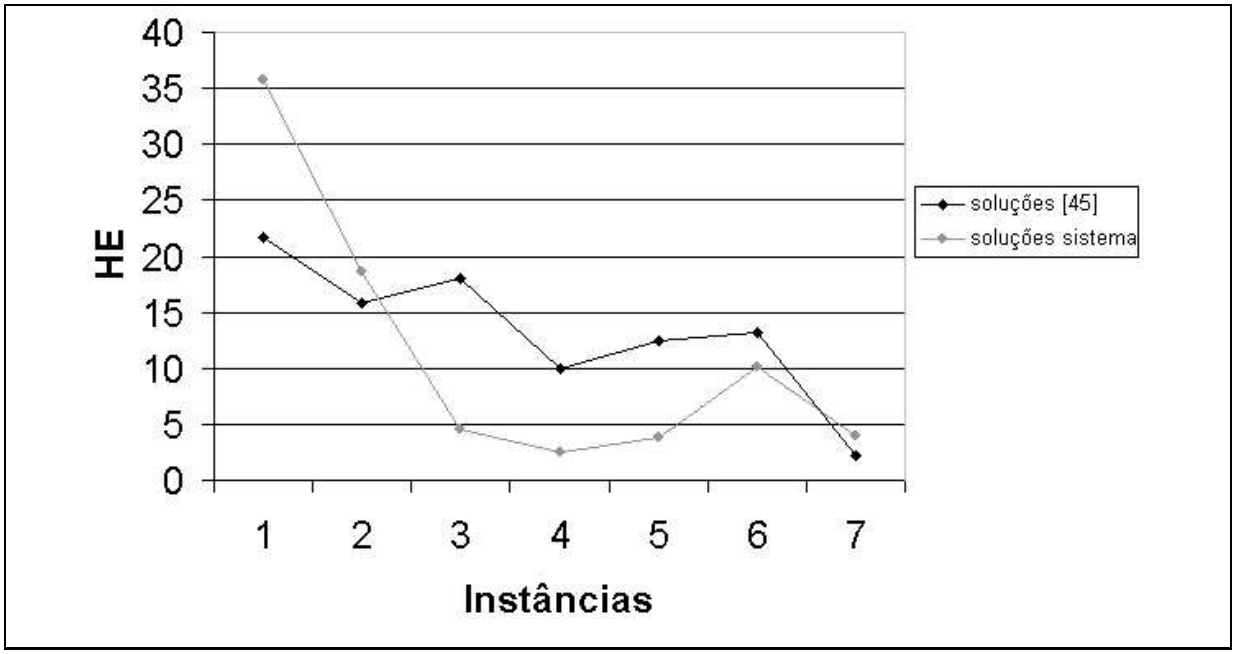


Figura 5.4: Comparativo na quantidade de horas extras entre as soluções de [45] e I_{10}^{400*}

Linha	DRM	DAM	DRm	DAm	CD
3	*	123	18.62%	13.97	3242.03
7	*	135	20.10%	15.71	3691.53

Linha	DRM	DAM	DRm	DAm	CD
3	6%	4	0.69%	0.44	803.8
7	30%	49	5.10%	3.78	2758.04

*Não há como calcular DRM porque a solução apresenta viagens partindo no mesmo minuto, ou seja, apresenta espaçamento entre viagens nulo

Tabela 5.20: Distribuição de horários nas soluções das empresas antes e depois da redistribuição

A título de ilustração, os horários de partida obtidos para a variação 1_{10}^{400} são mostrados nas tabelas 5.21 e 5.22. Nestas tabelas, a primeira linha indica a hora e as demais linhas indicam o minuto em que cada viagem parte. A primeira viagem do dia com origem no PC1, por exemplo, começa às 04:27. Partem também viagens às 05:05, 05:16, 05:28, e assim por diante.

Horários de Partidas no PC1

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
				27	05	01	03	05	05	04	03	07	04	00	01	05	05	05	10	14	17	26	21	15
					16	05	09	12	15	14	10	15	13	10	10	17	15	15	26	33	45			
					28	09	14	18	25	23	17	23	22	20	19	29	25	25	41	52				
					39	13	19	25	35	34	24	32	31	30	28	41	35	36	56					
					50	17	24	31	44	44	31	39	41	41	37	53	45	46						
						22	30	38	54	54	38	47	50	51	46		55	56						
						26	35	44			45	55			55									
						31	40	51			52													
						35	44	57			59													
						40	49																	
						44	54																	
						49	59																	
						54																		
						58																		

Tabela 5.21: Distribuição de horários no PC1 para a variação 1_{10}^{400}

Horários de Partidas no PC2

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
				22	13	07	05	07	02	00	00	01	02	01	04	05	05	06	09	07	11	06	14	13
					46	20	10	16	07	12	12	07	22	08	14	13	14	16	24	21	40	26		
						33	15	26	13	24	24	14	42	16	25	22	22	26	38	36		45		
						46	21	35	19	36	36	21		24	35	31	31	36	53	50				
						59	26	44	24	48	49	27		32	45	39	39	46						
							31	53	30			34		39	55	48	48	56						
							36		36			41		47		57	57							
							40		42			47		55										
							45		48			54												
							50		54															
							54																	
							59																	

Tabela 5.22: Distribuição de horários no PC2 para a variação 1_{10}^{400}

4. *Quantidade de viagens*

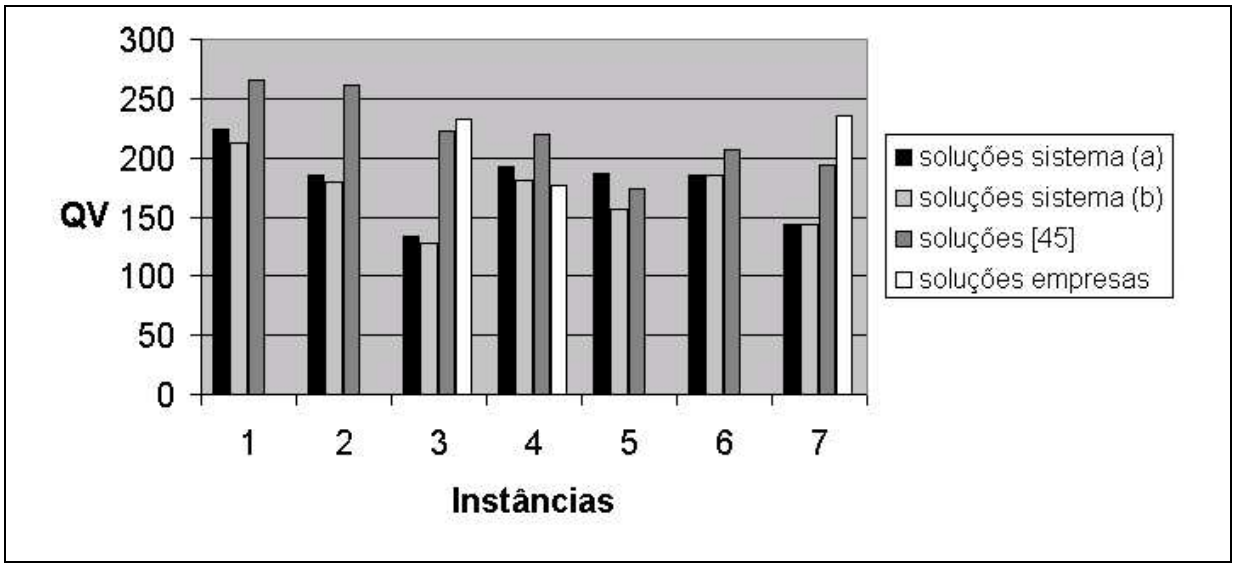


Figura 5.5: Quantidade de viagens nas soluções das empresas e de [45], e nas variações I_{10}^{400} e I_{10}^{400*}

A quantidade de viagens entre PCs efetuadas durante o dia é um indicativo dos recursos envolvidos na operação de uma linha de ônibus. Desde que a demanda seja atendida dentro do padrão de qualidade exigido, e que não haja um empilhamento maior nos PCs, é preferível efetuar menos viagens pois, além de se economizar combustível, contribui-se para diminuir o tráfego nas vias públicas. Embora a solução do sistema para a linha circular, i.e., a linha 4, tenha exigido 9% mais viagens do

que a solução da empresa, para as demais instâncias ocorreu o inverso. As soluções do sistema apresentaram redução de até 42% no número de viagens em relação às soluções das empresas e às geradas em [45]. A figura 5.5 traz uma comparação na quantidade de viagens entre as soluções das empresas, as soluções de [45] e as soluções das variações I_{10}^{400}(a) e I_{10}^{400*}(b). A ausência de uma barra na figura indica indisponibilidade de dados.

5. *Desempenho*

Todas as variações de instâncias foram resolvidas em menos de onze horas. Se considerada apenas a variação I_{10}^{400}, que foi tomada como referência, este tempo cai para pouco mais de nove horas. O sistema desenvolvido por [46] também retorna respostas por volta de algumas horas, mas apresenta melhor desempenho. De qualquer modo, os tempos de processamento são aceitáveis, uma vez que o tempo despendido na obtenção de uma solução manual é da ordem de vários dias. Através da figura 5.6, em que se exibe a média dos tempos de processamento para as variações de instância I_{10}^{400}, I_{3}^{400} e I_{10}^{200}, nota-se claramente que o gargalo do sistema é o escalonamento de motoristas. Apesar de o escalonamento de ônibus ser a etapa mais demorada na instância 3, o tempo para efetuá-lo nas diferentes variações das instâncias nunca ultrapassa uma hora.

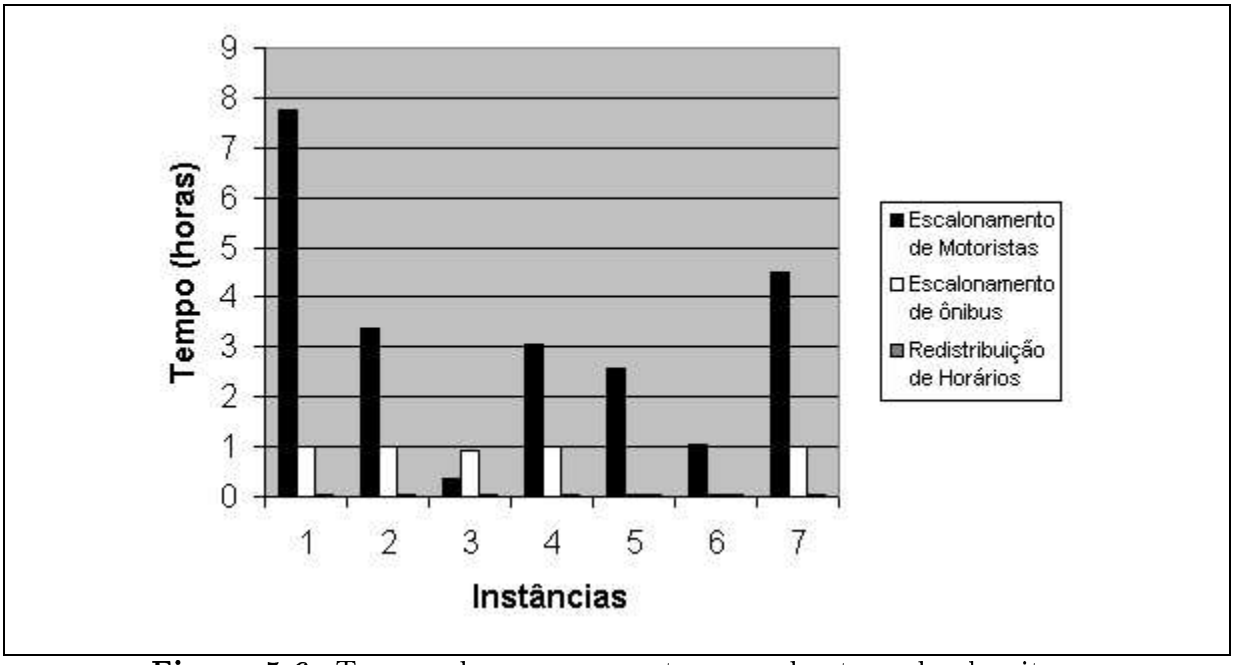


Figura 5.6: Tempos de processamento em cada etapa do algoritmo

6. *Padrão das atividades dos motoristas e dos ônibus*

   Grande parte dos algoritmos anteriormente propostos, assim como a abordagem utilizada em [46], constrói um conjunto grande de jornadas para depois selecionar um subconjunto que atenda à demanda da melhor forma possível. Assim, as atividades tendem a ser distribuídas de maneira equilibrada entre os motoristas e entre os ônibus, como é o caso das soluções de [45] e das soluções das empresas, em que as escalas também são construídas com foco nas atividades dos motoristas e dos ônibus. Os mapas de ônibus para estas soluções são exibidos nas figuras 5.7 e 5.8. O algoritmo proposto neste trabalho desvia-se desse paradigma, conforme pode ser observado na figura 5.9. Porém, isso não afeta o objetivo do sistema, que é utilizar o mínimo de recursos nas escalas construídas, atendendo estritamente à demanda e às restrições do problema.

   Nos mapas de ônibus, as viagens são representadas por retângulos brancos com diagonais descendentes, quando as viagem saem do PC1 e chegam no PC2, ou ascendentes, quando fazem o sentido contrário. As viagens entre um PC e a garagem, em qualquer dos sentidos, são ilustradas por retângulos pretos. Os retângulos cinzas representam descansos e rendições, estas em uma tonalidade mais escura. Um dos eixos do gráfico corresponde aos ônibus e o outro aos instantes do dia. A duração de cada atividade é proporcional ao tamanho do retângulo que a representa.

7. *Empilhamento máximo*

   Os empilhamentos máximos nas soluções geradas pelo sistema são menores do que nas soluções completas fornecidas pelas empresas, das instâncias 3 e 7. No entanto, são maiores em relação aos empilhamentos máximos nas soluções produzidas pelo sistema de [46]. Apesar de não ser rigidamente tratada no algoritmo proposto, somente nas instâncias 1 e 4, a restrição de empilhamento máximo é violada.

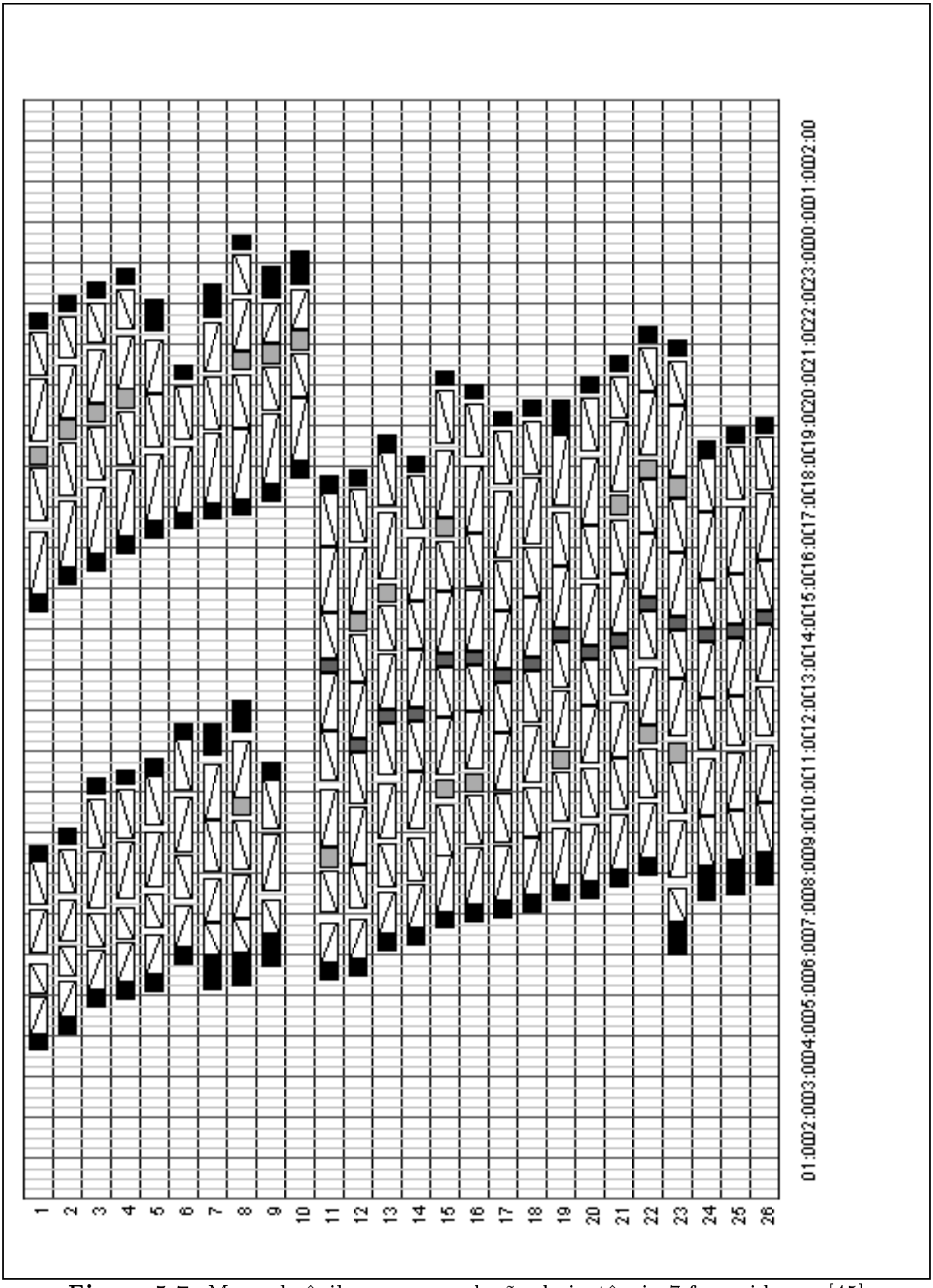


Figura 5.7: Mapa de ônibus para a solução da instância 7 fornecida em [45]

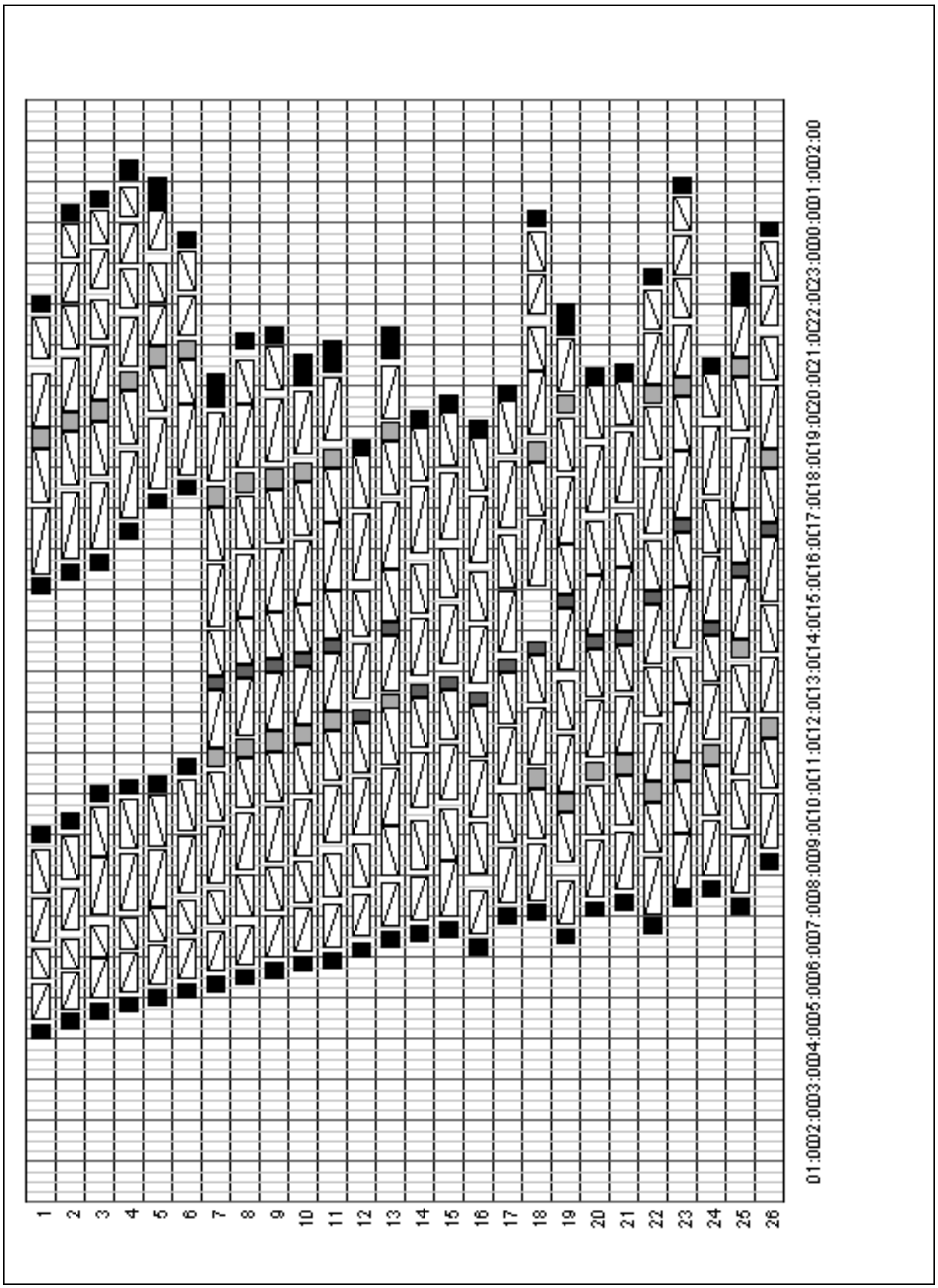


Figura 5.8: Mapa de ônibus para a solução da instância 7 fornecida pela empresa

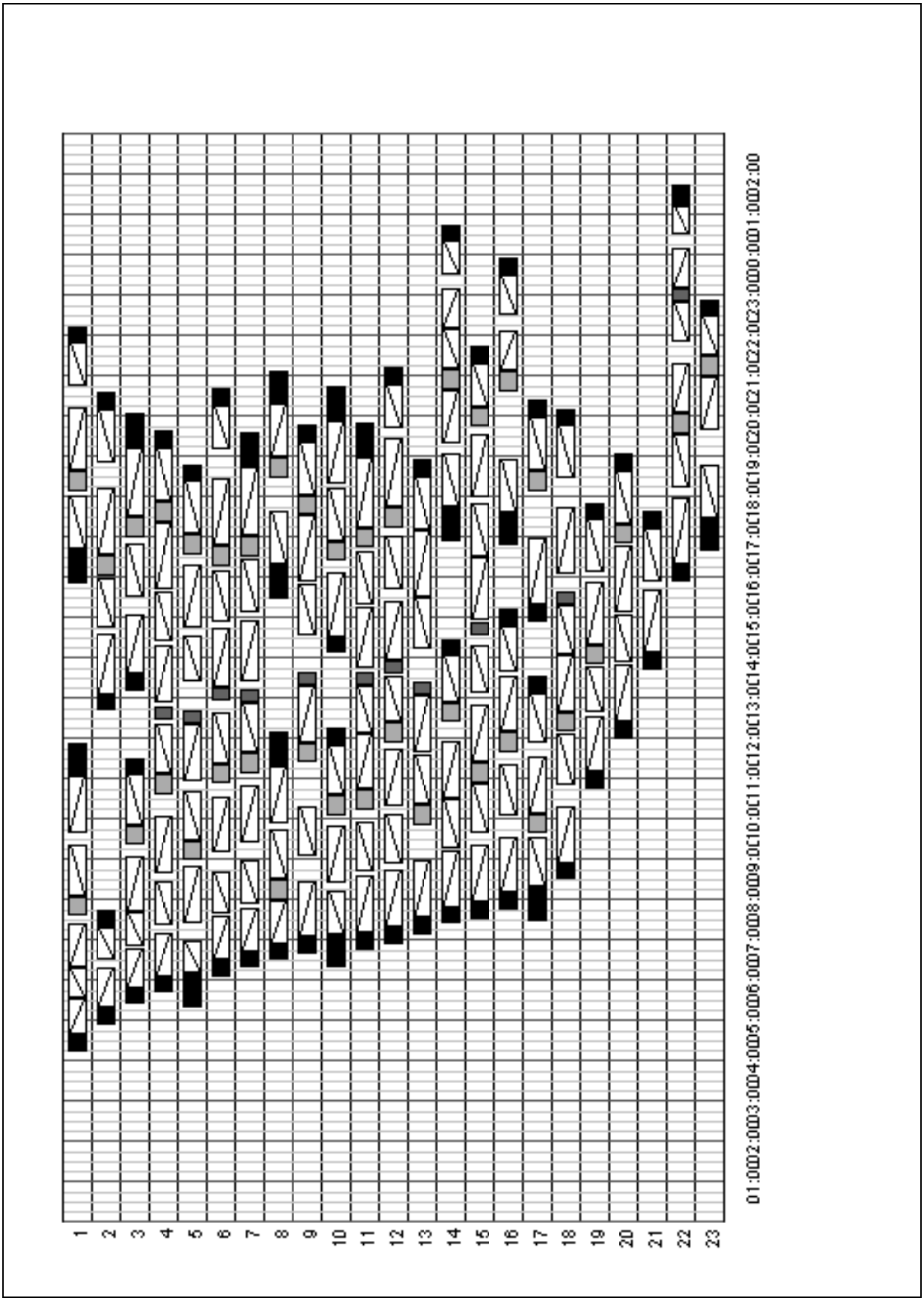


Figura 5.9: Mapa de ônibus para a solução da variação de instância 7_{10}^{400}

Capítulo 6

Conclusões

O principal objetivo deste trabalho foi resolver o PPV da melhor forma possível, utilizando para isso abordagens e modelos antes inexplorados no contexto do PPV. Basicamente, foram usadas e combinadas programação por restrições e programação linear. Obteve-se êxito nos três aspectos. A abordagem proposta demonstrou ser viável no cenário brasileiro, se não preferível. Os modelos construídos foram resolvidos eficientemente, apesar de reunirem formulações de restrições e de funções objetivos totalmente diferentes das que normalmente são utilizadas. E as técnicas comprovaram que se pode obter soluções de qualidade com desempenho adequado ao se resolver o PPV.

Se todos os sub-problemas do PPV pudessem ser resolvidos de forma integrada, provavelmente, esta seria a melhor abordagem. No entanto, integrá-los de tal forma que um mesmo modelo atenda às restrições dos escalonamentos de motoristas e de ônibus, além das restrições da tabela de horários, não é uma tarefa trivial, além de gerar modelos provavelmente intratáveis computacionalmente. A abordagem proposta, portanto, é seqüencial, mas as fases do algoritmo são dispostas de tal forma que permitem grande flexibilidade de modelagem para cada etapa, bom desempenho global e geração de soluções de qualidade.

No caso do escalonamento de motoristas, foi preciso formular cuidadosamente as restrições, devido a uma série de fatores. Como este sub-problema foi resolvido por uma combinação de programação por restrições e programação inteira, restrições lineares foram utilizadas para que pudessem estar presentes em ambos os modelos. A fim de gerar maior propagação, procurou-se utilizar restrições fortes e de aridades baixas, além de variáveis com domínios menores. Os modelos ainda foram concebidos tendo-se em mente a fácil identificação de simetrias e seu posterior tratamento. Também foram utilizadas heurísticas apoiadas em programação por restrições para determinar as instanciações parciais.

O escalonamento de veículos foi resolvido através da programação linear inteira pela

rapidez com que, nesta técnica, se constrói e se testa os modelos, e pela qualidade da solução gerada. Embora as soluções tenham sido obtidas dentro de um intervalo de tempo satisfatório, esperava-se que o desempenho, nesta etapa do algoritmo, fosse superior.

Para a redistribuição de horários, testou-se modelos de programação por restrições e de programação linear. No caso, a segunda técnica foi a que alcançou os melhores resultados em todos os aspectos.

Pode-se identificar quatro técnicas utilizadas neste trabalho: a programação linear sem a presença de variáveis inteiras, a programação linear inteira, a programação por restrições e outras técnicas heurísticas. Dentre elas, devido à sua eficiência e à qualidade da solução gerada, a primeira é normalmente indicada, desde que o problema possa ser satisfatoriamente representado por expressões lineares, e com um número de variáveis e restrições dentro dos limites tratáveis pelos resolvedores utilizados, como foi o caso da redistribuição de horários.

Quando variáveis inteiras são necessárias para modelar o problema, já não se tem garantia de uma resposta em tempo polinomial, mas a programação linear inteira pode, ainda, ser a melhor alternativa em várias situações. O modelo de programação inteira para o escalonamento de veículos, por exemplo, atendeu às necessidades do sistema, mas deixou um pouco a desejar quanto ao seu desempenho. Um algoritmo que combinasse heurísticas e programação por restrições poderia alcançar soluções mais rapidamente e, talvez, até de maior qualidade para este sub-problema do PPV. Porém, um tempo maior seria despendido para construí-lo, e soluções ótimas poderiam ainda não ser encontradas, devido ao porte dos modelos e a ineficiências nos algoritmos do resolvedor de programação por restrições. Por outro lado, esta técnica pode ser usada para tratar simetrias de forma eficiente. Também é bastante usada para descobrir soluções viáveis de problemas, principalmente em duas situações: quando o problema pode ser formulado como um problema conhecido mas é intratável por modelos exatos - neste caso, a combinação de programação por restrições com heurísticas é uma boa opção, e quando não se consegue enquadrar o problema em uma classe de modelos já conhecida [60].

Modelos mais complexos que envolvem muitas restrições e variáveis inteiras, como por exemplo, o modelo de escalonamento de motoristas aqui proposto, em geral, não são resolvidos por abordagens puras de programação inteira ou de programação por restrições. Algoritmos híbridos têm obtido sucesso nestes casos. Neste trabalho, foi introduzido, além dessas técnicas, o uso de heurísticas, representadas no sistema implementado principalmente pelas instanciações parciais. A programação por restrições dá suporte à implementação de heurísticas bem fundamentadas nas características específicas dos problemas.

Os resultados alcançados pelo sistema implementado nesse trabalho foram muito bons, conforme mostra o capítulo 5. Na maioria das instâncias, foram obtidas soluções melhores que as soluções geradas pelas empresas e pelo sistema desenvolvido por [46]. Em parte,

tal sucesso deve-se ao fato de o sistema ter sido construído especificamente para tratar o problema do transporte urbano brasileiro, ao contrário de outras ferramentas, citadas no capítulo 3, que foram simplesmente adaptadas.

Como a abordagem utilizada neste trabalho para resolver o PPV inova em muitos aspectos, várias direções de estudo podem ser desenvolvidas para que os modelos propostos atendam a outras restrições e se melhore o desempenho global e a qualidade das soluções geradas. Algumas sugestões são fornecidas na próxima seção.

Acredita-se que este trabalho pode ajudar no avanço das pesquisas relacionadas ao planejamento operacional das empresas de transporte coletivo e estimular o uso da combinação das técnicas de programação por restrições e de programação linear na construção de ferramentas computacionais para resolvê-los.

6.1 Trabalhos Futuros

Várias vertentes de pesquisa podem ser desenvolvidas para melhorar os resultados obtidos no presente trabalho:

- No cenário em que um motorista pode utilizar mais de um carro - há cidades brasileiras onde isso é permitido - a abordagem proposta pode apresentar resultados ainda melhores, já que cada jornada pode ser dividida em unidades menores. Neste caso, o escalonamento de ônibus provavelmente geraria escalas que utilizam frotas menores. Além disso, o empilhamento máximo deve ser reduzido, já que um ônibus não ficaria obrigatoriamente estacionado no PC durante o descanso de um motorista.

- Incluir as restrições que não foram tratadas no sistema, citadas na seção 1.2.4.

- O algoritmo 4.1, de escalonamento de motoristas, procura iniciar a escala utilizando um motorista. Caso não seja possível, procura uma escala com dois. Novamente, se não for possível, tenta com três, e assim por diante. No entanto, é possível calcular uma quantidade mínima de motoristas necessária para efetuar as viagens de uma linha. Assim, sabendo-se que são necessários pelo menos n motoristas, o algoritmo poderia iniciar buscando uma escala com n motoristas, poupando, desta forma, o tempo de processamento utilizado para descobrir que não havia escalas de motoristas possíveis para 1, 2, ..., e $n-1$ motoristas.

- Uma atenção especial foi depositada nos algoritmos de escalonamento de motoristas e de redistribuição de horários. Ao escalonamento de ônibus, entretanto, reservou-se pouco tempo por ser aparentemente o mais fácil. Outros algoritmos poderiam ser

construídos para tratar o escalonamento de ônibus, a fim de se alcançar melhores resultados, principalmente quanto ao seu desempenho. Uma sugestão é a combinação de programação por restrições com heurísticas.

- As linhas testadas utilizam um ou dois PCs. Uma variação do sistema poderia ser construída para tratar instâncias com mais de dois PCs. Também, convencionou-se que cada ônibus cobre uma ou duas jornadas de motoristas, mas com algumas alterações no modelo de escalonamento de ônibus, pode-se construir escalas em que mais jornadas possam ser cobertas por um mesmo ônibus.

- O algoritmo de redistribuição de horários mantém a ordem em que as viagens partem de cada PC, o que pode levar ao descarte de soluções possivelmente melhores. Um novo modelo em que isto não ocorra obrigatoriamente poderia ser desenvolvido.

- A tabela inicial de horários sempre segue a distribuição ideal mínima. No entanto, isso pode dificultar a construção de melhores escalas de motoristas e de ônibus. Um motorista pode ficar impedido de efetuar determinada viagem por causa de um ou dois minutos devido a alguma restrição, como por exemplo a referente à duração de viagens. Situações desta natureza poderiam ser contornadas com uma pequena flexibilização de certas restrições, ou então, com uma distribuição inicial de horários mais adequada.

- As soluções dos modelos de programação inteira para o escalonamento de motoristas não são aproveitadas na busca de soluções para modelos posteriores. Como os modelos para uma determinada instanciação parcial são muito parecidos, o aproveitamento de soluções anteriores é um recurso que pode ser explorado a fim de se alcançar um melhor desempenho.

- As soluções geradas apresentaram muitas jornadas curtas, que levam a um tempo ocioso elevado. Diferentes maneiras de se reduzir esse tempo ocioso poderiam ser pesquisadas. A penalização de jornadas curtas, por exemplo, poderia fazer com que jornadas assim fossem evitadas. Uma outra maneira de se reduzir o tempo ocioso seria inserir viagens em jornadas curtas, fazendo com que passem a ter duração superior a uma fronteira estabelecida.

Referências Bibliográficas

[1] E. Aarts e J. K. Lenstra, editores. *Local Search in Combinatorial Optmization*. John Wiley and Sons, 1997.

[2] A. B. Baker. *Intelligent Backtracking on Constraint Satisfaction Problems: Experimental and Theoretical Results*. Tese de Doutorado, University of Oregon, 1995.

[3] M. S. Bazaraa, J. J. Jarvis, e H. D. Sherali. *Linear Programming and Network Flows*. John Wiley and Sons, 1990.

[4] R. Bosch e M. Trick. Constraint programming and hybrid formulations for life. In *Proceedings of Formul'01 Workshop, CP'01*. 2001. Disponível em http://www.dcs.gla.ac.uk/~pat/cp2001/papers/.

[5] A. Ceder. Minimum cost vehicle scheduling with different types of transit vehicles. In J. R. Daduna, I. Branco, e J. M. P. Paixão, editores, *Computer-Aided Transit Scheduling*, volume 430 de *Lecture Notes in Economics and Mathematical Systems*, páginas 102–114. Springer, 1993.

[6] R. Clement e A. Wren. Greedy genetic algorithms, optimizing mutation and bus driver scheduling. In J. R. Daduna, I. Branco, e J. M. P. Paixão, editores, *Computer-Aided Transit Scheduling*, volume 430 de *Lecture Notes in Economics and Mathematical Systems*, páginas 213–235. Springer, 1993.

[7] A. Costa, I. Branco, e J. M. P. Paixão. Vehicle scheduling problem with multiple type of vehicles and a single depot. In J. R. Daduna, I. Branco, e J. M. P. Paixão, editores, *Computer-Aided Transit Scheduling*, volume 430 de *Lecture Notes in Economics and Mathematical Systems*, páginas 115–129. Springer, 1993.

[8] S. D. Curtis. *Constraint Satisfaction Approaches to Bus Driver Scheduling*. Tese de Doutorado, School of Computer Studies, University of Leeds, 2000.

[9] S.D. Curtis, B.M. Smith, e A. Wren. Forming bus driver schedules using constraint programming. Relatório Técnico 1999.05, School of Computer Studies, University of Leeds, 1999.

[10] J. R. Daduna e J. M. P. Paixão. Vehicle scheduling for public mass transit - An overview. In J. R. Daduna, I. Branco, e J. M. P. Paixão, editores, *Computer-Aided Transit Scheduling*, volume 430 de *Lecture Notes in Economics and Mathematical Systems*, páginas 76–90. Springer, 1993.

[11] K. Darby-Dowman e J. Little. Properties of some combinatorial optimisation problems and their effect on the performance of integer programming and constraint logic programming. *INFORMS Journal of Computing*, 10(3):276–286, 1998.

[12] M. Desrochers e F. Soumis. A column generation approach to the urban transit crew scheduling problem. *Transportation Science*, 23(1):1–13, 1989.

[13] ECRC. ECLiPSe. Disponível em http://www.icparc.ic.ac.uk/eclipse/Leaflet.doc.

[14] ECRC. *ECLiPSe 5.0 Library Manual*, 2001.

[15] M.A.N. Azevedo Filho, R.S.K. Kwan, e A. Wren. A alocação de ônibus e motoristas no Brasil: Alguma experiência prática. *Anais do VIII Congresso de Pesquisa e Ensino em Transportes*, 2:231–242, 1994.

[16] P. Flener, A.M. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, J. Pearson, e T. Walsh. Symmetry in matrix models. In *Proceedings of the CP'01 Workshop on Symmetry in Constraints*, páginas 41–48. 2001.

[17] P. Flener, A.M. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, e T. Walsh. Matrix modelling. In *Proceedings of Formul'01 Workshop, CP'01*. 2001. Disponível em http://www.dcs.gla.ac.uk/~pat/cp2001/papers/.

[18] S. Fores e Les Proll. Driver scheduling by integer linear programming - The TRACS II approach. Relatório Técnico 1998.01, School of Computer Studies, University of Leeds, 1998.

[19] S. Fores, Les Proll, e A. Wren. A column generation approach to bus driver scheduling. Relatório Técnico 1996.22, School of Computer Studies, University of Leeds, 1996.

[20] S. Fores, Les Proll, e A. Wren. TRACS to better schedules. Relatório Técnico 2001.03, School of Computing, University of Leeds, 2001.

[21] R. Fourer, D. M. Gay, e B. W. Kernigham. *AMPL: A Modeling Language For Mathematical Programming*. The Scientific Press, 1993.

[22] R. Freling. *Models and Techniques for Integrating Vehicle and Crew Scheduling*. Tese de Doutorado, Tinbergen Institute, Erasmus University, 1997.

[23] R. Freling, D. Huisman, e A. P. M. Wagelmans. Models and algorithms for integration of vehicle and crew scheduling. Relatório Técnico ERS-2000-14-LIS, Erasmus Research Institute of Management (ERIM), Rotterdam School of Management, 2000.

[24] T. Fruhwirth, A. Herold, V. Kuchenhoff, T. Le Provost, P. Lim, E. Monfroy, e M. Wallace. Constraint logic programming - an informal introduction. Relatório Técnico ECRC-93-5, ECRC, 1993.

[25] I. P. Gent e B. M. Smith. Symmetry breaking during search in constraint programming. Relatório Técnico 1999.02, School of Computer Studies, University of Leeds, 1999.

[26] C. Gervet. Large combinatorial optimization problems: a methodology for hybrid models and solutions. Apresentação convidada em JFPLC'98, 1998.

[27] N. Guerinik e M.V. Caneghem. Solving crew scheduling problems by constraint programming. In U. Montanari e F. Rossi, editores, *Principles and Practice of Constraint Programming - CP'95*, páginas 481–498. Springer, 1995.

[28] ILOG. *ILOG AMPL CPLEX System 7.0 User's Guide*, 2000.

[29] ILOG. *ILOG CPLEX 7.0 User's Manual*, 2000.

[30] R.S.K. Kwan e M.A. Rahin. Object oriented bus vehicle scheduling - the BOOST system. Relatório Técnico 1997.28, School of Computer Studies, University of Leeds, 1997.

[31] K. Marriott e P. J. Stuckey. *Programming with Constraints: An introduction*. MIT Press, 1998.

[32] S. Mayerle e J. A. Cruz. Escala de motoristas e alocação de frota no transporte urbano de passageiros. Programa de Pós-Graduação em Engenharia de Produção, Universidade Federal de Santa Catarina, 1997.

[33] M. Meier. Better late than never. Relatório Técnico ECRC-95-8, ECRC, 1995.

[34] M. Meier e J. Schimpf. Control in ECLiPSe. Relatório Técnico ECRC-95-7, ECRC, 1995.

[35] O. Mekkaoui, A. Palma, e R. Lindsey. Optimal bus timetables and trip timing preferences. Relatório Técnico 2000-11, Thema, Université de Cergy-Pontoise, 2000.

[36] T. Müller. Solving set partitioning problems with constraint programming. In *PAPPACT98*, páginas 313–332. The Practical Application Company Ltd, 1998.

[37] J. M. Paixão e I. Branco. A quasi-assignment algorithm for bus scheduling. *Networks*, 17:249–269, 1987.

[38] J. M. Paixão e I. Branco. Bus scheduling with a fixed number of vehicles. In J. R. Daduna e A. Wren, editores, *Computer-Aided Transit Scheduling: Proceedings of the Fourth International Workshop*, páginas 28–40. Springer Verlag, 1988.

[39] A. Palma e R. Lindsey. Optimal timetables for public transportation. Relatório Técnico 2000-09, Thema, Université de Cergy-Pontoise, 2000.

[40] C. H. Papadimitriou e K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, 1998.

[41] L. Proll e B. M. Smith. ILP and constraint programming approaches to a template design problem. Relatório Técnico 1997.16, School of Computer Studies, University of Leeds, 1997.

[42] P. Prosser e E. Selensky. On the encoding of constraint satisfaction problems with 0/1 variables. In *Proceedings of Formul'01 Workshop, CP'01*. 2001. Disponível em http://www.dcs.gla.ac.uk/~pat/cp2001/papers/.

[43] C. R. Reeves. *Modern Heuristic Techniques for Combinatorial Problems*. Wiley, 1993.

[44] R. Rodosek, M. Wallace, e T. Hajian. A new approach to integrate mixed integer programming with CLP. In *CP'96 Workshop on Constraint Programming Applications: An Inventory and Taxonomy*, páginas 45–54. 1996.

[45] M. M. Rodrigues. Comunicação particular.

[46] M. M. Rodrigues. *Problema de Planejamento de Viagens no Transporte Coletivo Urbano*. Tese de Mestrado, Instituto de Computação, UNICAMP, 2001.

[47] J. Schimpf, K. Shen, e M. Wallace. *Tutorial on Search in ECLiPSe*, 2001. Disponível em http://wwwhome.cs.utwente.nl/~etalle/lp/eclipse-doc/tutorial/tutorial.html.

[48] B. M. Smith. A tutorial on constraint programming. Relatório Técnico 1995.14, School of Computer Studies, University of Leeds, 1995.

[49] B. M. Smith. Reducing symmetry in a combinatorial design problem. Relatório Técnico 2001.01, School of Computing, University of Leeds, 2001.

[50] B. M. Smith, S. C. Brailsford, P. M. Hubbard, e H. P. Williams. The progressive party problem: Integer linear programming and constraint programming compared. Relatório Técnico 1995.08, School of Computer Studies, University of Leeds, 1995.

[51] F. A. Vanini. Projeto FAPESP tipo PIPE 00/13210-3.

[52] M. Wallace. Constraint programming. Relatório Técnico, IC-Parc, Imperial College, 1995. Disponível em http://www-icparc.doc.ic.ac.uk/papers.html.

[53] M. Wallace. Survey: Practical applications of constraint programming. Relatório Técnico, IC-Parc, Imperial College, 1995. Disponível em http://www-icparc.doc.ic.ac.uk/papers.html.

[54] M. Wallace. Constraint logic programming: Status and prospects. In *Proceedings of The Third World Congress on Expert Systems*, páginas 13–23. 1996.

[55] M. Wallace, S. Novello, e J. Schimpf. ECLiPSe: A platform for constraint logic programming. Relatório Técnico, IC-Parc, Imperial College, 1997. Disponível em http://www-icparc.doc.ic.ac.uk/papers.html.

[56] L. A. Wolsey. *Integer Programming*. John Wiley and Sons, 1998.

[57] A. Wren. Heuristics ancient and modern: Transport scheduling through the ages. *Journal of Heuristics*, 4:87–100, 1998.

[58] A. Wren e N. D. F. Gualda. Integrated scheduling of buses and drivers. Relatório Técnico 1997.32, School of Computer Studies, University of Leeds, 1997.

[59] A. Wren e J. Rousseau. Bus driver scheduling - an overview. In J. R. Daduna, I. Branco, e J. M. P. Paixão, editores, *Computer-Aided Transit Scheduling*, volume 430 de *Lecture Notes in Economics and Mathematical Systems*, páginas 173–187. Springer, 1993.

[60] T. H. Yunes. *Problemas de Escalonamento no Transporte Coletivo: Programação por Restrições e Outras Técnicas*. Tese de Mestrado, Instituto de Computação, UNICAMP, 2000.

[61] T. H. Yunes, A. V. Moura, e C. C. de Souza. Solving large scale crew scheduling problems with constraint programming and integer programming. Relatório Técnico IC-99-19, Instituto de Computação, UNICAMP, 1999.