Heurísticas Busca Tabu para o Problema de Programação de Tripulações de Ônibus Urbano

Euler Horta Marinho

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre. Área de concentração: Otimização e Inteligência Artificial.

Orientador: Luiz Satoru Ochi
Co-orientador: Marcone Jamilson Freitas Souza

Niterói, Maio de 2005.

Heurísticas Busca Tabu para o Problema de Programação de Tripulações de Ônibus Urbano

Euler Horta Marinho

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre. Área de concentração: Otimização e Inteligência Artificial.

Aprovada por:

Prof. Luiz Satoru Ochi / IC-UFF (Presidente)

Prof. Marcone Jamilson Freitas Souza / DECOM-UFOP

Profa. Lúcia Maria de Assumpção Drummond / IC-UFF

Prof. Eduardo Uchoa / EP-UFF

Prof. Fábio Protti / IM-UFRJ

Niterói, Maio de 2005.

Ficha Catalográfica elaborada pela Biblioteca da Escola de Engenharia e Instituto de Computação da UFF

M338 Marinho, Euler Horta
Heurísticas busca tabu para o problema de programação de tripulações de ônibus urbano / Euler Horta Marinho. – Niterói,. RJ : [s.n.], 2005.
xx f.

Orientador: Luiz Satoru Ochi.
Dissertação (Mestrado em Ciência da Computação) - Universidade Federal Fluminense, 2005.

1. Metaheurística. 2. Busca Tabu. 3. Transporte urbano. I. Título.

CDD 005.136

"Apliquei então meu espírito ao esclarecimento de tudo isso: os justos, os sábios e seus atos estão nas mãos de Deus. O homem ignora se isso será amor ou ódio. Tudo é possível. Um mesmo destino para todos: há uma sorte idêntica para o justo e para o ímpio, para aquele que é bom como para aquele que é impuro, para o que oferece sacrifícios como para o que deles se abstém. O homem bom é tratado como o pecador e o perjuro como o que respeita o seu juramento.

Entre tudo o que se faz debaixo do sol, é uma desgraça só existir para todos um mesmo destino: por isso o espírito dos homens transborda de malícia, a loucura ocupa o coração deles, durante a vida, depois da qual vão para a casa dos mortos. Porque, enquanto um homem permanece entre os vivos, há esperança: mais vale um cão vivo que um leão morto. Com efeito os vivos sabem que hão de morrer, mas os mortos não sabem mais nada; para eles não há mais recompensa, porque sua lembrança está esquecida. Amor, ódio, ciúme, tudo já pereceu; não terão mais parte alguma, para o futuro, no que se faz debaixo do sol.

Ora, pois, come alegremente teu pão e bebe contente teu vinho, porque Deus já apreciou teus trabalhos. Traja sempre vestes brancas e haja sempre azeite (perfurmado) em tua cabeça. Desfruta da vida com a mulher que amas, durante todos os dias da fugitiva e vã existência que Deus te concede debaixo do sol. Esta é a tua parte na vida, o prêmio do labor a que te entregas embaixo do sol. Tudo o que tua mão encontra para fazer, faze-o com todas as tuas faculdades, pois que na região dos mortos, para onde vais, não há mais trabalho, nem ciência, nem inteligência, nem sabedoria."

Ecle. 9,1-10

Agradecimentos

Foram muitas as pessoas que contribuiram para a realização desse trabalho. Este é dedicado a cada uma delas que por algum motivo, e de alguma maneira, estiveram presentes nessa via crucis magnífica.

Acima de tudo, agradeço a Deus por ter me dado a saúde e a fé necessários para a conclusão dessa jornada.

Agradeço especialmente aos professores Luiz Satoru Ochi, Lúcia Maria de Assumpção Drummond e Marcone Jamilson Freitas Souza, cuja orientação e auxílio começaram ainda nos tempos da graduação na UFOP e prosseguiram pelo mestrado. À professora Lucília Camarão de Figueiredo do Departamento de Computação/UFOP, pelos recursos que auxiliaram no desenvolvimento deste projeto. Ao professor Gustavo Peixoto Silva do Departamento de Computação/UFOP que foi o responsável pela solução da Programação de Veículos, problema preliminar ao abordado neste trabalho. A vocês o meu Muito Obrigado por toda compreensão, apoio e incentivo à realização desse curso.

Agradeço aos meus pais, Paulo Luciano Marinho e Geralda Joana Marinho, pela dedicação e preocupações que os acompanharam em cada dia.

Aos meus familiares, especialmente aos meus irmãos Eliane, Elisabete e Eugênio.

À Flávia Dias Marques pela amizade, carinho, amor, compreensão e estímulo, presentes em cada dia, sem os quais tudo seria muito mais difícil. Agradeço ainda à Fúlvia e à Dona Maria Joaquina pelas orações.

Aos amigos da "Mouseville", conterrâneos de UFOP, pelo acolhimento desde os meus primeiros dias em Niterói.

Ao amigo Luiz Henrique de Campos Merschmann pela companhia e pela convivência nesta luta. Ao amigo gaúcho Haroldo Gambini Santos pelos mesmos motivos e ainda pela introdução e salvação no Mundo Linux.

Agradeço ainda a todos as pessoas que estiveram perto nessa caminhada. Aos colegas da graduação que, como eu, se aventuraram em desbravar a terra estrangeira da cidade de

Niterói. Aos novos colegas, do estado do Rio, Filipe Quintieri e Luís Carlos Retondaro.

Meus sinceros e humildes agradecimentos a todos vocês.

Resumo da Tese apresentada à UFF como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação (M.Sc.)

Heurísticas Busca Tabu para o Problema de
Programação de Tripulações de Ônibus Urbano

Euler Horta Marinho

Maio/2005

Orientador: Luiz Satoru Ochi
Co-orientador: Marcone Jamilson Freitas Souza
Programa de Pós-Graduação em Computação

O Problema de Programação de Tripulações de ônibus urbano (PPT) trata da geração de escalas de trabalho de motoristas e cobradores. Neste, as viagens programadas para a empresa devem ser atribuídas às tripulações com o menor custo possível. A resolução do PPT é de grande importância, uma vez que nos gastos totais de uma empresa a mão-de-obra operacional representa um dos maiores custos. O PPT pertence a uma classe de problemas combinatoriais chamada programação de pessoal (*staff scheduling ou rostering*). Tais problemas são de difícil resolução, uma vez que apresentam um grande número de restrições operacionais e trabalhistas. Além disso, pertencem à classe de problemas NP-difíceis, uma vez que o problema de particionamento de conjuntos (*Set Partitioning Problem*) e o problema de recobrimento de conjuntos (*Set Covering Problem*) podem ser reduzidos a esse tipo de problema. Sendo assim, o uso exclusivo de métodos exatos limita-se a resolver problemas de pequenas dimensões. Para problemas de dimensões maiores, a abordagem mais comum é através de heurísticas. Neste trabalho foi proposta a aplicação da metaheurística Busca Tabu ao PPT de uma empresa de transporte público do município de Belo Horizonte-MG. Primeiramente, desenvolveu-se uma versão denominada BT-FI-IC, a qual foi comparada com um algoritmo da literatura baseado no Método de Pesquisa em Vizinhança Variável (VNS-RTL). Os resultados obtidos demonstraram a superioridade desta versão, tanto em relação à qualidade de solução final, quanto em relação à sua rapidez na produção de soluções de boa qualidade. Com o propósito de comparar a Busca Tabu com um método exato foi utilizado um método de geração completa de colunas para a resolução do PPT. Para tal, foram criadas 11 instâncias a partir de uma instância fornecida pela empresa. O método BT-FI-IC foi então aprimorado com um mecanismo de relaxação adaptativa (BTAR-FI-IC) e com um mecanismo de reconexão por caminhos (BTPR-FI-IC). As soluções obtidas pelas três versões da Busca Tabu foram então comparadas às soluções ótimas e à melhor solução conhecida para a instância da empresa. O método BTAR-FI-IC apresentou-se como o mais eficiente na abordagem do problema, mostrando-se robusto e capaz de produzir rapidamente soluções de qualidade.

Abstract of Thesis presented to UFF as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

Tabu Search heuristics to the Bus Driver Scheduling Problem

Euler Horta Marinho
Maio/2005

Advisor: Luiz Satoru Ochi
Co-advisor: Marcone Jamilson Freitas Souza
Department: Computer Science

The Bus Driver Scheduling Problem (BDSP) attempts to setup work schedules for drivers and conductors, with the best possible utilization of resources. The solving of BDSP is a subject of great relevance. It is due to the fact that most of operational costs involved in the process have come from labor costs. The BDSP belongs to a class of combinatorial problems called staff scheduling or rostering. Such problems are difficult to solve, since a large number of labor and operational rules should be observed. Moreover, they belong to the class NP-Hard, since the Set Covering and Set Partitioning problems can be reduced to this type of problem. Thus, the exclusive use of exact methods is limited to solving models with a small number of different restrictions. For realistic models, with a large number of different restrictions, one common approach is to use heuristic techniques. In this work a solution method, based on the Tabu Search metaheuristic, was proposed for solving the BDSP for a Brazilian company of public transport from Belo Horizonte city. Firstly, a strategy named BT-FI-IC was implemented and compared to an algorithm from literature based on Variable Neighborhood Search method (VNS-RTL). The results had demonstrated the superiority of the BT-FI-IC strategy in relation to quality of the final solution and speed in producing good quality solutions. With the aim of compare the Tabu Search to a mathematical programming method, a complete column generation approach was implemented to solve the BDSP. For this, eleven artificially created instances, with smaller dimensions were created from the real problem. The BT-FI-IC method was improved with an Adaptive Relaxation mechanism (BTAR-FI-IC) and with a Path Relinking mechanism (BTPR-FI-IC). The solutions obtained through the three strategies were compared to the optimal solutions and also to the best known solution for the real problem. The BTAR-FI-IC method was the most efficient one, showing robustness and ability to produce good solutions quickly.

Palavras-chave

1. Programação de Tripulações de Ônibus Urbano
2. Busca Tabu
3. Relaxação Adaptativa
4. Reconexão por Caminhos
5. Metaheurísticas

Sumário

Resumo **iv**

Abstract **v**

Lista de Figuras **x**

Lista de Tabelas **xii**

1 Introdução **1**

2 Revisão Bibliográfica **3**

2.1 Introdução 3

2.2 Programação de tripulações 4

2.2.1 Programação de tripulações de aviões 5

2.2.2 Programação de tripulações de ônibus 9

2.2.3 Programação de tripulações de trens 16

2.3 Problema genérico de programação de tripulações 18

2.3.1 Formulação matemática 18

2.4 Método VNS-RTL 21

2.5 Busca Tabu 24

2.6 Módulos adicionais na Busca Tabu 26

2.6.1 Estratégias de intensificação 26

2.6.1.1 Reinício da BT com soluções elite 26

2.6.1.2 Freqüência das soluções elite 26

2.6.1.3 Memória e intensificação 26

2.6.2 Estratégias de diversificação 27

2.6.2.1 Freqüência das soluções não-elite 27
2.6.2.2 Busca Tabu Reativa 27
2.6.2.3 Abordagens por *Ejection Chains* 27
2.6.3 Reconexão por Caminhos 28
2.6.4 Relaxação Adaptativa 31

3 O Problema de Programação de Tripulações de Ônibus Urbano 34

4 Método de Solução do PPT 39

4.1 Representação 39
4.2 Estruturas de Vizinhança 41
4.3 Função de Avaliação 41
4.4 Procedimento de geração da solução inicial (IC) 43
4.5 Algoritmos propostos 44
4.5.1 Busca Tabu com primeira melhora (BT-FI-IC) 44
4.5.2 Busca Tabu com melhor vizinho (BT-BI-IC) 45
4.5.3 Busca Tabu com primeira melhora e Relaxação Adaptativa (BTAR-FI-IC) 45
4.5.4 Busca Tabu com primeira melhora e Reconexão por Caminhos (BTPR-FI-IC) 47
4.6 Método de Geração Completa de Colunas 48

5 Resultados e Discussão 50

5.1 Instâncias-teste 51
5.2 Comparação com o Método VNS-RTL 52
5.3 Resolução das instâncias por geração completa de colunas 56
5.4 Avaliação das versões BT-FI-IC e BTAR-FI-IC 56
5.5 Distribuição de probabilidade empírica para as versões BT-FI-IC e BTAR-FI-IC 61
5.6 Comparação entre as versões BTAR-FI-IC e BTPR-FI-IC 64

6 Conclusões e trabalhos futuros 68

Referências 71

Lista de Figuras

1 O planejamento de transporte 4

2 Visualização gráfica do PGPT 19

3 Método de Pesquisa em Vizinhança Variável 21

4 Estruturas de Vizinhança do método VNS-RTL 22

5 Algoritmo de Busca Tabu 25

6 Algoritmo de Busca Tabu com Reconexão por Caminhos 30

7 Procedimento de Reconexão por Caminhos 31

8 Oscilação Estratégica 32

9 Bloco de um veículo particionado em tarefas 35

10 Solução para o Problema de Programação de Tripulações 38

11 Modelagem de uma tarefa 39

12 Modelagem de uma jornada 40

13 Representação da solução do PPT 41

14 Estruturas de Vizinhança 42

15 Algoritmo de Busca Tabu com Relaxação Adaptativa 46

16 O mecanismo de intensificação por reconexão por caminhos 47

17 Evolução dos algoritmos BT-FI e VNS-RTL nos instantes iniciais do refinamento 54

18 Distribuição de probabilidade empírica de encontrar um valor alvo em função do tempo para a instância PPT-Enterprise 55

19 Comportamento da função de penalização para o tempo ocioso 58

20 Distribuição de probabilidade empírica de encontrar um valor alvo em função do tempo para a instância PPT-Enterprise 61

21 Distribuição de probabilidade empírica de encontrar um valor alvo em função do tempo para a instância PPT-L2104-94 62

22 Distribuição de probabilidade empírica de encontrar um valor alvo em função do tempo para a instância PPT-L4150-63 63

23 Distribuição de probabilidade empírica de encontrar um valor alvo em função do tempo para a instância PPT-Enterprise 67

Lista de Tabelas

1 Melhor solução obtida pelo método VNS-RTL 24

2 Características dos algoritmos 51

3 Características das instâncias 52

4 Número de colunas geradas por instância 52

5 Penalizações dos requisitos não essenciais para comparação com o método VNS-RTL 53

6 Penalizações dos requisitos essenciais para comparação com o método VNS-RTL 53

7 Desempenho dos algoritmos BT-BI-IC e BT-FI-IC em relação ao método VNS-RTL 53

8 Características das melhores soluções geradas pelos algoritmos 54

9 Solução ótima para cada instância e o tempo para encontrá-la 56

10 Valores das constantes $\sigma_{i,min}$ e $\sigma_{i,max}$ para cada restrição 57

11 Penalizações dos requisitos essenciais 57

12 Penalizações dos requisitos não essenciais 57

13 Tempo de execução para cada instância 58

14 Distância média das melhores soluções conhecidas, para cada instância, nas versões BT-FI-IC e BTAR-FI-IC 59

15 Resultados médios em execuções com limites de tempo fixo para as versões BT-FI-IC e BTAR-FI-IC 60

16 Tempo de execução para cada instância 64

17 Distância média das melhores soluções conhecidas, para cada instância, nas versões BTAR-FI-IC e BTPR-FI-IC 65

18 Resultados médios em execuções com limites de tempo fixo para as versões BTAR-FI-IC e BTPR-FI-IC 66

1 Introdução

O Problema de Programação de Tripulações de Ônibus Urbano (PPT) também conhecido como *Crew Scheduling Problem* ou *Bus driver Scheduling Problem* trata da geração de escalas de trabalho de motoristas e cobradores de empresas de transporte público. Este problema tem sido objeto de estudo desde a década de 60, sendo Elias (1964) o trabalho pioneiro. Essa constante investigação deve-se às contínuas transformações ocorridas nos sistemas de transporte público onde as empresas deste setor buscam a redução de custos operacionais e o aprimoramento de seus processos produtivos, mantendo a qualidade de seus produtos e/ou serviços.

A despeito disso, pouco tem sido feito no Brasil para o aperfeiçoamento dos sistemas de transporte urbano (ANTP, 1999). O modelo de desenvolvimento centrado no transporte rodoviário provocou um desbalanceamento no transporte de pessoas e mercadorias no país com conseqüências negativas como congestionamentos crônicos nas vias urbanas, queda da mobilidade e da acessibilidade, degradação das condições ambientais e altos índices de acidentes de trânsito. Segundo a ANTP (1999), uma política de transporte urbano é essencial para garantir melhores condições de deslocamento de mercadorias e pessoas. Para tal finalidade, a melhoria da qualidade dos serviços deve ocorrer no âmbito tecnológico (veículos), gerencial (capacitação dos operadores), de atendimento (cobertura física e temporal), de integração, e de informação ao usuário. O aumento da eficiência de operação do sistema depende da garantia de espaço viário e condições adequadas de trânsito, na forma de faixas e vias exclusivas, apoiadas por sistemas de sinalização e controle eficientes. Além disso, para que ocorra a democratização do acesso ao transporte público, deve haver a garantia de tarifas toleráveis para os usuários, o que depende em grande parte do aumento da eficiência e redução de custos por parte das organizações do setor. Como a mão-de-obra é uma das componentes que mais pesam na planilha de custos, uma pequena redução neste item pode significar um ganho considerável no custo total (BOUZADA, 2002), o que justifica qualquer trabalho no sentido de minimizar os custos com a mão-de-obra.

Embora o PPT tenha sido largamente estudado e aplicado em outros países como o Canadá (BLAIS; LAMONT; ROUSSEAU, 1990), a Inglaterra (FORES; PROLL; WREN, 1999; KWAN; KWAN; WREN, 2001), e Portugal (LOURENÇO; PAIXÃO; PORTUGAL,

2001), suas técnicas de resolução são ainda pouco difundidas e raramente aplicadas à nossa realidade. Isso se deve, em parte, pelo estágio primário em que se encontram as empresas de transporte público no Brasil, onde as empresas desse setor normalmente geram manualmente as escalas de trabalho.

O desenvolvimento de um método de solução eficiente para o PPT é, portanto, de grande importância. Neste trabalho aborda-se o PPT fazendo-se um estudo de caso de uma empresa que opera no sistema de transporte público do município de Belo Horizonte-MG.

Este trabalho está organizado como segue. O presente Capítulo introduz o problema de programação de tripulações e ressalta a importância de resolvê-lo eficientemente. No Capítulo 2 faz-se uma revisão bibliográfica sobre o problema de programação de tripulações, bem como os problemas à ele correlatos, e suas principais técnicas de solução. No Capítulo 3 explica-se, com detalhes, o problema de programação de tripulações abordado. No Capítulo 4 apresenta-se a metodologia utilizada para a abordagem do problema tratado. No Capítulo 5 mostra-se e discute-se os resultados computacionais. No Capítulo 6 faz-se a conclusão e aponta-se os direcionamentos futuros.

2 Revisão Bibliográfica

2.1 Introdução

A programação de pessoal (*Staff Scheduling ou Rostering*) é o processo de construir escalas de trabalho para funcionários de forma que uma organização possa satisfazer a demanda por seus bens ou serviços. Primeiramente, deve-se determinar o número de funcionários, com qualificações próprias, necessários para corresponder à demanda pelos serviços. Cada funcionário será alocado a um turno (*shift*) para que, em diferentes períodos do dia, exista a satisfação da referida demanda, e jornadas (*duties*) são associadas a cada turno. Todos os acordos das convenções coletivas de trabalho devem ser observados durante esse processo (ERNST et al., 2004).

É extremamente difícil encontrar boas soluções para problemas como esse, complexos e repletos de restrições, e ainda mais difícil determinar a solução de menor custo que corresponda às preferências de empregados, distribua turnos uniformes de trabalho entre os funcionários e satisfaça todas as restrições trabalhistas.

Essa dificuldade é devida ao grande número de combinações, o que inviabiliza uma resolução por enumeração explícita de todas as soluções possíveis. Essa intratabilidade computacional se deve ao fato de o problema de programação de pessoal ser NP-difícil, uma vez que o problema de particionamento de conjuntos (*Set Partitioning Problem*) e o problema de recobrimento de conjuntos (*Set Covering Problem*), que são NP-difíceis (GAREY; JOHNSON, 1979), podem ser reduzidos a esse tipo de problema.

Sendo assim, o uso exclusivo de técnicas de programação matemática, ditas exatas, limita-se a resolver problemas de pequenas dimensões. Para problemas de dimensões maiores, a abordagem mais comum é através de heurísticas. Em geral, as características próprias de diferentes indústrias e organizações determinarão quais modelos matemáticos e algoritmos específicos para a programação de pessoal devem ser desenvolvidos. Existe uma ampla bibliografia sobre construções de escalas de trabalho para funcionários, utilizando técnicas distintas para sua obtenção. Neste trabalho, são citadas somente algumas das principais publicações sobre este assunto na área de sistemas de transporte.

2.2 Programação de tripulações

Na área de transportes aéreos, férreos, metroviários, rodoviários e urbanos, a programação de pessoal (*Staff Scheduling*) é conhecida como programação de tripulações (*Crew scheduling and rostering*).

As características únicas dessa área de aplicação da programação de pessoal são (ERNST et al., 2004):

- Estão envolvidas características temporais e espaciais, isto é, cada tarefa é caracterizada por um horário de início e localização e um horário de término e localização.
- Todas as tarefas a serem executadas pelos empregados são provenientes de uma dada tabela de horários (malha de vôos, no caso de aviões; bloco dos veículos, no caso de ônibus). Tarefas são pequenos elementos (ou blocos de construção) obtidos da tabela de horários de aviões, trens, metrôs ou ônibus. Uma tarefa pode ser um trecho de vôo em linhas aéreas, uma viagem entre dois ou mais segmentos numa jornada de trem, ou uma viagem entre duas ou mais paradas consecutivas numa linha de ônibus.

A Figura 1, utilizada em Lourenço, Paixão & Portugal (2001), situa a programação de tripulações dentro do processo de planejamento de transporte.

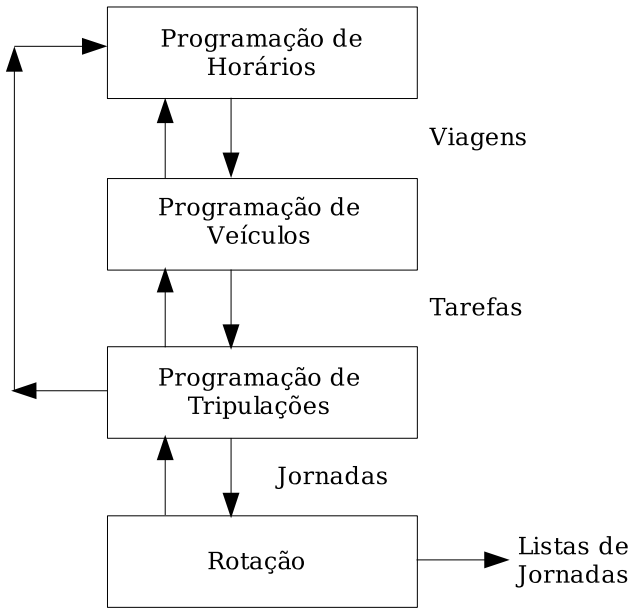

Figura 1: O planejamento de transporte

Como se observa, o planejamento de transporte é decomposto em vários subproblemas devido a sua complexidade: programação de horários, programação de veículos,

programação de tripulações e rotação (*roster scheduling*) (FRELING, 1997).

O serviço de transporte é composto por um conjunto de linhas, usualmente identificadas por um número, que corresponde ao meio de transporte deslocando de uma localização a outra. Para a determinação das linhas, são definidas rotas que mostram a direção do fluxo de pessoas/mercadorias. Para cada linha, a respectiva freqüência é determinada com base na demanda e nível de serviço. Após, uma tabela de horários é construída, resultando em viagens que correspondem a uma localização inicial e uma localização final, e a um horário de início e um horário de término. A programação de veículos associa veículos a viagens. O problema de programação de tripulações gera jornadas para as tripulações. E, a rotação constrói jornadas, em geral mensais, compreendendo as jornadas resultantes da fase anterior juntamente com folgas, feriados, férias das tripulações e restrições trabalhistas.

Nas seções seguintes serão considerados, separadamente, os diferentes setores do mercado de transporte.

2.2.1 Programação de tripulações de aviões

Devido a sua importância e impacto econômicos, a programação de tripulações de linhas aéreas (*airline crew scheduling and rostering*) é provavelmente a maior área de aplicação da programação de pessoal (ERNST et al., 2004). Muitos trabalhos têm contribuído com metodologias e aplicações nessa área.

O objetivo na programação de tripulações de linhas aéreas é minimizar o custo de execução da escala de vôo resultante, sujeito a restrições tais como:

- Cada trecho de vôo precisa ser coberto por apenas uma tripulação, embora viagens mortas (que consistem em transportar apenas tripulações como passageiros e visam ao reposicionamente de aeronaves) sejam permitidas em algumas versões do problema;
- Cada chave de vôo (*pairing* ou rotação) precisa começar na base da tripulação, prosseguir em torno do sistema, e retornar à mesma base;
- Cada chave de vôo precisa estar em conformidade com as limitações das *Agências de Regulação da Aviação* e com as regras trabalhistas e operacionais da companhia aérea;
- O número de tarefas em cada base deve estar compreendido entre os limites mínimos e máximos, de acordo com o planejamento de mão-de-obra da companhia aérea.

A programação de tripulações varia entre diferentes linhas aéreas, especialmente entre linhas aéreas na América do Norte e Europa. As maiores diferenças são decorrentes

de categorias de tripulações, tipos de vôos, regularidade de vôos, regras operacionais e trabalhistas. Essas diferenças são ressaltadas em Andersson et al. (1997).

Lavoie, Minoux & Odier (1988) apresentam um algoritmo para a abordagem por recobrimento de conjuntos (SCP) e geração parcial de todas as possíveis colunas. A geração de colunas é baseada no cálculo do caminho mínimo sobre um grafo de factibilidade acíclico. Se a geração de colunas falhar na geração de uma escala candidata atraente, o SCP é resolvido para a otimalidade via o algoritmo *branch-and-bound*. Foram utilizadas instâncias reais para teste envolvendo de 20 a 329 trechos de vôo. Os resultados foram em torno de 10% melhores do que o melhor resultado obtido pelos planejadores da programação de tripulações da empresa.

Gershkoff (1989) discute a programação de tripulações na American Airlines. O autor usou um modelo de programação linear inteira (particionamento de conjuntos), onde as linhas representam trechos de vôo a serem cobertos e as colunas, chaves de vôo candidatas que cobrem um ou mais trechos de vôo. Se a chave de vôo j cobre o trecho i, então o elemento para a i-ésima linha e j-ésima coluna será igual a 1; caso contrário, 0. Cada coluna apresenta um custo associado. O conjunto ótimo de chaves de vôo é representado pelo conjunto de colunas cujo custo total é minimizado. O procedimento de otimização, utilizando um procedimento heurístico, seleciona de um conjunto inicial de chaves de vôo, o qual é uma solução factível, um subconjunto de chaves de vôo. Aquelas chaves não selecionadas são temporariamente descartadas da análise. Dado este subconjunto, o procedimento gera todas as possíveis colunas para os segmentos (linhas) que fazem parte deste. A técnica *branch-and-bound* é utilizada para encontrar um novo subconjunto de chaves a partir do subconjunto original. Se este sobconjunto possui um custo menor que o subconjunto original então uma melhora foi encontrada e as novas chaves de vôo substituem as antigas na solução corrente. Essas novas chaves, junto com as chaves não envolvidas na análise, se tornam a nova solução inicial. O processo é repetido até que uma condição de parada seja atingida. O procedimento alcança resultados ótimos para problemas de pequeno a médio porte (até 200 vôos por dia).

Anbil et al. (1991) apresentam novas abordagens para a programação de tripulações na American Airlines. Neste trabalho é usado um modelo de programação linear inteira (particionamento de conjuntos) que é posteriormente relaxado via Relaxação Lagrangeana. Basicamente a idéia é determinar rapidamente um bom conjunto de multiplicadores lagrangeanos e iniciar a programação linear a partir desse ponto. Na geração do subconjunto de chaves a ser analisado, uma chave de vôo é descartada se o seu custo exceder um determinado limiar. Dessa forma, subproblemas de maior porte (subconjuntos com até 25 chaves de vôo) podem ser tratados. Novos procedimentos também reduzem o impacto de mínimos locais (como aceitar movimentos que a princípio não reduzem o custo total da solução inicial).

Em Desaulniers et al. (1997) é abordada a programação de tripulações da Air France utilizando um modelo de fluxo em redes com recursos variáveis formulado como um problema de programação inteira, não linear e multi-produto. Devido à não-linearidade do modelo é aplicada uma extensão do princípio da decomposição de Dantzig-Wolfe. O modelo fica então reduzido a um problema de particionamento de conjuntos que é resolvido pela aplicação da técnica *branch-and-bound*. Foram utilizadas 11 instâncias extraídas a partir de dados reais dessa empresa. As dimensões dos problemas-teste variavam entre 154 e 1157 trechos de vôo. Os resultados obtidos foram comparados com aqueles apresentados pelo sistema de programação de tripulações pré-existente na Air France. Em média as soluções obtidas pela abordagem descrita foram 6,24% melhores.

A programação de tripulações da Rio-Sul Linhas Aéreas é abordada em Cabral et al. (2000) e em Pimentel (2005). No primeiro trabalho, os autores utilizam a abordagem heurística para solução de problemas de empacotamento *bin-packing*, guiada por modelos de programação matemática, os quais consistem em dois procedimentos: o primeiro tenta garantir que todas as alocações de tarefas sejam feitas, obedecendo às regras operacionais e trabalhistas; o segundo tenta realizar diversas otimizações na escala original, com o objetivo de alcançar uma boa distribuição de *duties*. Na fase de alocação de tarefas, há duas estratégias de seleção. Na primeira estratégia, seleciona-se o membro da tripulação mais conveniente para receber tarefas. Nesse caso, são consideradas inicialmente as tripulações que apresentam maiores impedimentos para recebimento de tarefas. O algoritmo trata posteriormente as tripulações que possuem maior tempo de trabalho disponível. Na segunda estratégia, seleciona-se as tarefas prioritárias para alocação. Um modelo matemático controla a segunda estratégia de seleção. A função objetivo é composta por dois termos: a duração da tarefa (considera-se como prioritárias as tarefas maiores) e pela avaliação de quanto é crítica a tarefa para a escala como um todo. Na fase de otimização procura-se uma distribuição balanceada da carga de trabalho (em quilometragem voada e horas trabalhadas) entre os membros da tripulação. Os problemas usados para teste consistem em três rotações, sendo 2 para pilotos e uma para atendentes de vôo. Para uma das rotações de pilotos, o método exato encontrou a melhor solução em 3 horas de processamento, e não conseguiu melhorá-la nas 48 horas seguintes. Para as outras duas rotações o método exato sequer encontrou uma solução viável. Para avaliar a performance do método heurístico proposto, foram comparadas as soluções exata e heurística para o problema onde foi encontrada solução. O método exato encontrou a melhor solução (sub-ótima) com um *gap* de 4% em relação à solução ótima do problema linear relaxado correspondente, enquanto que no modelo heurístico proposto esse *gap* foi de 2%. Além disso, no método heurístico a solução final foi encontrada em apenas 48 minutos.

Emden-Weinert & Proksch (1999) apresentam uma abordagem para a programação de tripulações via *Simulated Annealing* (SA). Os autores ilustram como melhorar a performance do SA no que diz respeito ao tempo de execução e à qualidade da solução final. Para

reduzir o tempo de execução, foram testadas três heurísticas de construção de soluções iniciais e comparados os tempos de execução do SA, em diferentes temperaturas iniciais, utilizando as três heurísticas. A heurística que levava em consideração as características do problema apresentou, em média, os melhores resultados. O uso de procedimentos de busca local conjugados com o SA possibilitou a melhora da qualidade da solução final. Foi implementada a paralelização do SA em múltiplas execuções independentes, na tentativa de melhorar a qualidade da solução final. Essa estratégia mostrou-se mais eficiente à medida que os tempos de execução de cada SA são maiores (através da diminuição da taxa de resfriamento). Foram utilizadas 2 instâncias para testes, com 1395 e 4612 vôos por mês, provenientes de dados da empresa Lufthansa Systems Berlin.

Em Ozdemir & Mohan (2001) a programação de tripulações é abordada via algoritmo genético. O problema é modelado como um grafo de vôo. Neste caso, os vértices representam vôos enquanto as arestas representam restrições de dependência entre os vôos. Existe uma aresta entre o vértice representando o vôo X para o vértice representando o vôo Y se: (i) Y parte da cidade destino de X e (ii) se Y parte após um intervalo de tempo específico após a chegada de X. Cada caminho no grafo representa uma seqüência factível de vôos que pode ser associada a uma tripulação. A execução do algoritmo genético é seguida pela aplicação de uma busca local. O algoritmo foi testado em 24 instâncias provenientes de dados reais e comparado aos algoritmos baseados na Heurística de Chvatal, Heurística de Bäck e com um algoritmo genético aplicado à uma abordagem por recobrimento de conjuntos. O algoritmo se mostrou melhor que as outras abordagens em 17 instâncias.

Pimentel (2005) propõe uma nova metodologia para a solução de problemas de recobrimento de conjuntos de grande porte, com aplicação ao problema de alocação de tripulações de companhias aéreas. A autora apresenta uma formulação que divide a solução do problema em duas etapas: geração das rotações e otimização. A abordagem utilizada envolve a prévia geração de um conjunto de rotações, e a partir deste, a busca de uma solução inteira. Esta abordagem permite que, na etapa de geração de rotações sejam tratadas questões específicas do problema, como função de custo das rotações e restrições da regulamentação. Na etapa de otimização é desenvolvida uma metodologia mais geral para resolver problemas que podem ser modelados como um problema de recobrimento de conjuntos. A geração das rotações é feita por meio de um algoritmo paralelo, baseado na metaheurística GRASP. Nesta etapa, são utilizados grafos de jornadas de serviço, que são conjuntos de trechos de vôos a serem realizados durante a jornada de trabalho dos tripulantes, obedecendo a diversas restrições impostas pela regulamentação. Com a utilização da metaheurística GRASP foi possível obter conjuntos de rotações de boa qualidade, evitando a geração de um número excessivo de rotações, bem como a aplicação de técnicas de geração de colunas, que seriam inviáveis para os recursos computacionais disponíveis. Para a fase de otimização foi utilizado o modelo de recobrimento de conjuntos, para o qual foi desenvolvido um algoritmo híbrido, com base em técnicas de *branch-and-bound*.

A escolha do modelo de recobrimento de conjuntos se deve ao fato que para esse modelo é mais fácil encontrar soluções viáveis e tratar viagens mortas (tripulantes se deslocando como passageiros). O algoritmo obteve boas soluções para problemas reais de construção de rotações.

2.2.2 Programação de tripulações de ônibus

A programação de tripulações de ônibus (*bus driver scheduling*) tem por objetivo encontrar o conjunto de jornadas diárias de trabalho, de menor custo, que cobre todas as viagens ou blocos de veículos de uma empresa de ônibus. O bloco do veículo é o itinerário de um veículo partindo da garagem e retornando à mesma. Esse bloco pode ser particionado em tarefas, de forma que a partição ocorra somente em uma oportunidade de troca, ou seja, um horário e uma localização nos quais a troca de tripulações seja possível. A jornada é um conjunto de tarefas que podem ser associadas a uma tripulação. Para cada jornada, um custo é associado, o qual representa custos reais, como horas extras, horas noturnas e custos de alimentação, e custos artificiais como por exemplo, troca de veículos, tipo de serviço, número de horas trabalhadas em relação à média. Outros componentes do custo podem ser considerados se os mesmos forem requisitados pela empresa de transporte.

O trabalho a ser realizado pelas tripulações está sujeito a várias regras dependentes da legislação local e de acordos das convenções coletivas de trabalho que, em geral, são:

- O tempo máximo de trabalho das tripulações;
- O tempo máximo de trabalho sem intervalo para alimentação;
- A duração dos intervalos para repouso ou alimentação.

A programação de tripulações de ônibus tem recebido considerável atenção em sistemas de transporte público. O progresso feito nessa área é investigado numa série de volumes intitulados *Computer-Aided Scheduling of Public Transport*: Wren (1981), Rousseau (1985), Daduna & Wren (1988), Desrochers & Rousseau (1992), Daduna, Branco & Paixão (1995), Wilson (1999), Daduna & Voss (2001).

Sousa (1991) apresenta uma abordagem interativa para a programação de tripulações, ou seja, o usuário pode manualmente alterar as soluções geradas pelo sistema para que correspondam às suas expectativas. Para a criação das jornadas foi usada uma heurística simples derivada daquela utilizada em Parker & Smith (1981). Essa heurística realiza a varredura dos blocos dos veículos em busca de tarefas (porções de trabalho) factíveis (que tenham uma duração mínima especificada pelo usuário). No início e no fim de cada tarefa deve existir tempo para troca de tripulações. As tarefas factíveis são iterativamente associadas a jornadas. Posteriormente tenta-se combinar as demais tarefas que permaneceram

não alocadas de modo a torná-las factíveis. O processo é repetido até que todas as tarefas tenham sido alocadas ou até que o tempo para execução tenha terminado. Deve ser observado que essa heurística pode produzir soluções incompletas (isto é, no fim da execução podem existir tarefas não alocadas), as quais devem ser completadas manualmente. Testes foram realizados com 3 instâncias reais (blocos dos veículos). Os resultados foram comparados com a confecção manual para a programação das tripulações. Para a primeira instância o tempo de execução foi de 20 minutos (manual: 8 horas), para a segunda 30 minutos (manual: 16 horas) e para a terceira 1 hora e 40 minutos (manual: 24 horas). Essa abordagem constitui um módulo que faz parte de um sistema de planejamento de transporte.

O sistema HASTUS é ilustrado em Blais, Lamont & Rousseau (1990). Este sistema é composto de três módulos denominados HASTUS-Bus, HASTUS-Macro e HASTUS-Micro. O primeiro faz a programação de veículos. O segundo é usado como um passo inicial para a programação de tripulações. E o último produz escalas de trabalho detalhadas, utilizando a solução parcial do HASTUS-Macro como um guia. Abaixo serão descritos os módulos responsáveis pela programação de tripulações.

A primeira entrada para o HASTUS-Macro é o conjunto de blocos de veículos a serem cobertos. As regras trabalhistas constituem a segunda maior entrada. A saída do HASTUS-Macro consiste de uma lista de porções de trabalho que constituem a maneira menos custosa de cobrir os blocos de veículos. Essas porções de trabalho são descritas pelo seu horário de início e sua duração.

Em seguida, o módulo HASTUS-Micro é utilizado para produzir as tarefas detalhadas que serão alocadas às tripulações. Primeiramente, os blocos do veículos são efetivamente particionados tão próximo quanto possível da solução parcial do HASTUS-Macro. Os blocos são inicialmente particionados nas oportunidades de troca disponíveis para produzir uma solução inicial. Esta solução é melhorada, recortando um bloco após outro de uma maneira iterativa até a solução tornar-se estável. Durante este processo, a função objetivo a ser minimizada é o quadrado da diferença entre o número de tarefas de cada tipo na Micro e Macro soluções. Após, o HASTUS-Micro particiona os blocos dos veículos em tarefas factíveis, selecionando um caminho do começo ao fim do bloco sobre os arcos representando todos as tarefas potenciais que podem ser originadas a partir do bloco. Um algoritmo de caminho mínimo é utilizado para encontrar o melhor caminho sobre estes arcos. Este segundo passo é necessário para agregar as tarefas provenientes do procedimento de particionamento de blocos, enquanto respeita um número de restrições, incluindo tempo máximo de horas extras e tempo máximo de viagens entre oportunidades de troca. Este algoritmo é baseado numa formulação de fluxo em redes e gera soluções inteiras. A saída deste passo é um conjunto de jornadas de custo mínimo factíveis.

Em Fores, Proll & Wren (1999) é descrito o sistema TRACS II. As etapas de execução

do sistema podem ser sumarizadas como:

1. Calcular todas as possíveis oportunidades para tripulações viajarem como passageiros em táxis ou serviços fretados entre as oportunidades de troca (somente nos problemas onde o trajeto a pé não é apropriado);

2. Gerar um grande conjunto de jornadas diárias potenciais, cada uma satisfazendo todas as restrições legais juntamente com regras que inibem a construção de jornadas indesejadas;

3. Se apropriado, filtrar inteligentemente algumas jornadas que não desempenham uma tarefa significativa em cobrir o trabalho do veículo;

4. Agregar conjuntos de jornadas geradas de acordo com diferentes parâmetros, se apropriado;

5. Resolver um modelo de recobrimento de conjuntos, com extensões para limitar o número de jornadas de diferentes tipos. Primeiramente é utilizada uma heurística construtiva para cobrir todo o trabalho dos veículos com um subconjunto de jornadas potenciais. Um problema de programação linear relaxado é resolvido utilizando uma técnica de geração de colunas especializada para instâncias de maior porte, e, caso contrário, uma abordagem de penalidade dual ascendente *dual steepest edge*. A programação linear relaxada provê um número de jornadas alvo;

6. A técnica *branch and bound* é utilizada sobre a solução relaxada para encontrar uma solução inteira com o número de jornadas alvo, se possível. Caso contrário o número de jornadas alvo é incrementado;

7. Qualquer sobrecobertura (jornadas cobrindo os mesmos trabalhos de veículos) é resolvida por um processo que pode ser automatizado.

Foram testadas 7 instâncias provenientes de dados reais. As duas estratégias de resolução do modelo de recobrimento de conjuntos (*dual steepest edge* e geração de colunas) foram comparadas. A primeira foi executada sobre um conjunto inicial de jornadas reduzido heuristicamente enquanto a outra, sobre um grande conjunto de jornadas previamente geradas. Em 5 das 7 instâncias o número de jornadas encontrado pela estratégia de geração de colunas foi menor do que aquele encontrado pela primeira estratégia. Isso inclui um problema onde nenhuma solução reduzida pode ser previamente encontrada e, logo, a primeira estratégia não pode ser executada. Apesar de haver um aumento do tempo de execução da geração de colunas comparado com a primeira estratégia, os tempos continuam aceitáveis pelo usuário. O sistema TRACS II alcança melhores resultados que estratégias baseadas em métodos de programação linear convencionais segundo Fores, Proll & Wren (1999).

Wren & Wren (1995) descrevem um estudo de viabilidade de um algoritmo genético com o objetivo de verificar a adequação de tal abordagem como um módulo do sistema IMPACS. Dado um grande conjunto de jornadas inicial, deve ser selecionado a partir deste, um subconjunto que cobre cada porção de trabalho dos veículos pelo menos uma vez, de maneira que o custo total seja minimizado e certas restrições na combinação das jornadas sejam satisfeitas. O cromossomo representa uma escala de trabalho. Cada posição do cromossomo (*locus*) representa uma porção de trabalho do veículo e cada alelo, uma jornada que cobre a respectiva porção de trabalho de sua posição. Foi desenvolvido um novo operador de cruzamento denominado *balls in the air*. Para a geração de um cromossomo filho os genes (jornadas) de cada cromossomo pai são colocados num conjunto em ordem cronológica. Desse conjunto é sorteado aleatoriamente, a cada vez, um gene para a formação do novo filho. Se for sorteado um gene que cobre apenas trabalhos já cobertos pelos outros genes correntes do filho ele é descartado do conjunto. Dessa forma são selecionados os genes suficientes para a geração de um novo filho. Foram executados testes com diversas configurações num pequeno problema para o qual uma solução de 17 jornadas havia sido encontrada. O algoritmo genético, em um dos testes, utilizando três pais para o cruzamento conseguiu obter uma solução com 15 jornadas após 941 iterações.

Em Kwan, Kwan & Wren (1999) é proposto um algoritmo genético para realizar a função da técnica *branch and bound* do sistema TRACS II descrito em Fores, Proll & Wren (1999). O algoritmo genético objetiva resolver o problema de programação de tripulações como um problema de recobrimento de conjuntos. O algoritmo genético proposto não produz ou trabalha com escalas completas. Uma heurística de construção tem por entrada informações da solução do algoritmo genético. Características especiais (bons atributos) passados através das gerações do algoritmo, são usados para alimentar a heurística de construção. Dessa forma a busca no espaço de soluções pode ser reduzida. Testes foram realizados com 7 instâncias (2 de ônibus e 5 de trens) provenientes de dados reais. As instâncias tinham de 10000 a 30000 jornadas potenciais e de 150 a 500 porções de trabalho de veículos. O algoritmo genético foi comparado à estratégia exata em termos do número de jornadas. Em três instâncias as duas abordagens foram equivalentes, em três instâncias o algoritmo genético foi pior por uma jornada a mais e em uma instância, pior por duas jornadas a mais. Li & Kwan (2000) apresentam um algoritmo genético com avaliação *fuzzy* abrangente. A escala é iterativamente construída por uma heurística gulosa. As jornadas são selecionadas a partir de um conjunto de jornadas potenciais por meio de critérios *fuzzy*. Esses critérios são baseados em funções sobre a estrutura e geralmente sobre a aptidão dos indivíduos. São atribuídos pesos a essas funções e as mesmas são combinadas de forma a se obter uma avaliação global. Embora seja possível obter um conjunto de pesos geral para uma classe de problemas, esses pesos foram ajustados a cada caso para uma melhor performance do algoritmo. O algoritmo genético proposto tem por objetivo encontrar o conjunto de pesos para um determinado problema. Foram utilizadas 11 instâncias (4 de

ônibus e 7 de trens) provenientes de dados reais para a avaliação do algoritmo. Como medida de qualidade das soluções obtidas foi usado o desvio padrão em relação à melhor solução conhecida. As soluções obtidas pelo algoritmo em termos de número de jornadas e custo total foram bem próximos as melhores soluções conhecidas e superiores àquelas obtidas por duas heurísticas gulosas (em que selecionava-se a cada passo a jornada que cobria mais porções de trabalho dos veículos) utilizadas para comparação. Em termos do custo da solução, o algoritmo foi apenas 1,62% pior que a melhor solução conhecida enquanto as duas heurísticas gulosas foram respectivamente 27,08% e 22,91% piores.

Em Shen & Kwan (2000) o sistema HACS (SHEN; KWAN, 2001) é estendido para lidar com problemas que apresentam Janelas de Oportunidade de Troca (JOTs). A maior parte dos sistemas automáticos de programação de tripulações, até então, podiam lidar apenas com oportunidades de troca em tempos discretos. As estruturas de vizinhança utilizadas foram as mesmas do trabalho anterior, com as devidas alterações para lidar com JOTs. Para a avaliação do sistema foram utilizadas 4 instâncias provenientes de dados reais e uma gerada artificialmente. Para a primeira classe de testes as instâncias foram expandidas de maneira a apresentar JOTs. Inicialmente foram comparados os testes envolvendo as instâncias com e sem JOTs. Nas instâncias com JOTs o sistema foi, em média, 2,16% melhor em termos de custo (horas totais trabalhadas) em relação às instâncias sem JOTs. Na segunda classe de testes foi testado o efeito das JOTs. Para tal propósito, a solução obtida pelo TRACS II (FORES; PROLL; WREN, 1999) para cada instância foi utilizada como solução inicial do HACS. Essa solução foi expandida para apresentar JOTs. Os resultados mostraram que o sistema pode sempre melhorar a melhor solução conhecida (SHEN; KWAN, 2000). Nesse caso, as soluções, em termos de custo, foram em média 2,26% melhores.

Wren, Kwan & Kwan (2000) propõem um algoritmo genético como um módulo do sistema TRACS II. Caso a técnica *branch and bound* não encontre uma solução viável, o algoritmo genético é executado utilizando o número de jornadas alvo da solução relaxada. Se a solução encontrada pelo algoritmo genético for maior que o número de jornadas alvo, a técnica *branch and bound* é executada novamente, tendo por número de jornadas alvo o número de jornadas da solução obtida pelo algoritmo genético. Se uma solução for encontrada, repete-se a aplicação da técnica com o número de jornadas alvo progressivamente menor, até que não seja possível obter uma solução com o número de jornadas alvo desejado. No final o sistema retorna a melhor solução obtida. Testes foram realizados com 15 instâncias provenientes de dados reais de empresas de ônibus e trens. O algoritmo genético foi comparado em relação à estratégia TRACS II original e à resolução manual em termos do número de jornadas. A resolução manual foi possível em 8 das 15 instâncias, sendo que em uma foram violadas algumas restricões trabalhistas. Para as instâncias em que uma solução manual foi encontrada, o algoritmo genético e o TRACS II foram, no mínimo, tão bons quanto ou freqüentemente superior àquela. Em duas instâncias o sis-

tema TRACS II não conseguiu obter solução viável. O algoritmo genético foi superior em duas instâncias de dimensões mais elevadas, as quais precisaram da aplicação de técnicas de decomposição (consistindo no particionamento do problema original em sub-problemas e resolver cada problema separadamente) para a resolução pelo TRACS II.

Em Kwan, Kwan & Wren (2001) é descrita uma abordagem híbrida na qual um algoritmo genético colabora com uma heurística gulosa para encontrar escalas de trabalho para tripulantes. O papel do algoritmo genético é derivar um bom conjunto de jornadas de trabalho para a heurística gulosa a partir de um conjunto de jornadas potenciais. Essas jornadas são selecionadas a partir de um grande conjunto de jornadas, por um método exato ou por uma heurística. A heurística gulosa por sua vez devolve a escala construída para o algoritmo genético para avaliação e extração de alguns atributos importantes encontrado na escala. Esses atributos representam forte conhecimento do domínio do problema, que são propagados e utilizados pelo algoritmo genético no processo evolutivo. Um importante atributo considerado é chamado cadeias de troca (*relief chains*). Essas cadeias consistem de um conjunto de oportunidades de troca. Cadeias de troca bem formadas, especialmente nos horários de pico, podem evitar o uso desnecessário de motoristas extras. Testes foram realizados com 20 instâncias (8 de ônibus e 12 de trens). Os resultados foram comparados com aqueles obtidos por resolução manual e com a técnica *branch and bound* utilizada por Fores, Proll & Wren (1999). Em três instâncias a técnica exata não obteve solução viável. Duas instâncias de dimensões mais elevadas precisaram de técnicas de decomposição para a resolução pela técnica exata. A resolução manual falhou em encontrar soluções viáveis para as duas instâncias de maior porte e para mais 5 outras instâncias. Em duas soluções obtidas pela resolução manual foram violadas algumas restrições trabalhistas.

Lourenço, Paixão & Portugal (2001) propõem a aplicação das metaheurísticas GRASP, Algoritmo Genético e Busca Tabu (BT) para o módulo de programação de tripulações do sistema GIST. O PPT é formulado como um problema de recobrimento de conjuntos. Para a solução deste, o conjunto de todas as colunas (ou jornadas) factíveis é gerado por enumeração completa, ou seja, todas as combinações de porções de trabalho dos veículos (linhas) que estejam de acordo com as regras operacionais e trabalhistas. O GRASP utiliza um mecanismo de oscilação estratégica. A BT é aplicada em quatro estágios cada qual considerando uma estrutura de vizinhança diferente. Para a BT ainda foi implementado uma estratégia de intensificação aplicada ao final de alguns estágios, baseada em um método de programação matemática para instâncias de menor dimensão e o próprio GRASP implementado, para instâncias de maior dimensão. O algoritmo genético utiliza um operador de *crossover* denominado *perfect offspring*. Este operador considera 2 pais e tenta obter o melhor filho desses pais por meio da solução de um subproblema de recobrimento de conjuntos, onde todas as linhas são consideradas mas somente as colunas presentes nos pais são levadas em consideração. Após o cruzamento, uma estratégia de

intensificação similar àquela da BT é aplicada. Os algoritmos foram testados sobre dados reais e comparados à solução obtida pela versão prévia do sistema GIST, a qual baseava-se em programação linear. Os resultados mostraram que as metaheurísticas Busca Tabu e Algoritmo Genético obtiveram bons resultados em tempos computacionais razoáveis em relação à versão baseada em programação linear.

Shen & Kwan (2001) utilizam Busca Tabu para a resolução do problema de programação de tripulações do sistema HACS. A abordagem é baseada numa representação do problema envolvendo seqüências de *links*. *Links* representam pares direcionados de oportunidades de troca (OTs) de chegada (*arrival*-ARO) e oportunidades de troca de partida (*departure*-ARO). Essas oportunidades de troca correspondem à primeira e à última OT de um *spell*, o qual representa o trabalho contínuo a ser operado por um tripulante sem intervalo. Os *links* e suas oportunidades de troca associadas compõem o espaço de soluções. No referido sistema, a Busca Tabu é aplicada em 4 fases sequenciais cada uma utilizando uma estrutura de vizinhança distinta, a saber: trocar dois *links*, trocar dois *spells*, inserir um *spell* em um *duty* e recortar blocos. Nesta última estrutura, porções de *spells* podem ser trocadas entre *duties*. Nos testes foram utilizadas 4 instâncias provenientes de dados reais. Os resultados foram comparados em relação àqueles obtidos pelos sistema TRACS II (FORES; PROLL; WREN, 1999) e pelo algoritmo genético descrito em Kwan, Kwan & Wren (1999). As soluções do TRACS II são as melhores soluções conhecidas para os problemas-teste. Em termos do número de *duties* as soluções obtidas pelo HACS foram em média 3,31% piores que a do TRACS II, sendo que para uma das instâncias os resultados foram iguais, e em relação ao algoritmo genético, 1,07% piores (em duas instâncias os resultados foram iguais e em uma o HACS foi 2,86% melhor).

Yunes (2000) aborda o problema de escalonamento de mão-de-obra oriundo da operação diária de uma empresa de ônibus urbano da cidade de Belo Horizonte. Foram tratados os problemas de programação de tripulações (*crew scheduling*), que trata a alocação diária de viagens a duplas de funcionários (motorista e cobrador), e de rotação (*crew rostering*), que parte da solução do problema anterior e constrói uma escala de trabalho para o período de um mês. Cada um desses problemas foram resolvidos utilizando-se técnicas de programação matemática e programação por restrições. Para o problema de *crew scheduling* a técnica de programação matemática se mostrou superior à técnica de programação por restrições. Para resolver problemas de maior porte foi desenvolvida uma heurística Lagrangeana. Para o problema de *crew rostering*, a programação por restrições apresentou melhores resultados, tanto em desempenho, quanto em qualidade de solução. Entretanto, nenhuma das técnicas isoladas foi capaz de tratar instâncias reais do problema. Adotando-se um algoritmo híbrido de geração de colunas combinando as duas técnicas mencionadas, foi possível encontrarem-se soluções ótimas para instâncias muito maiores.

Mauri (2005) apresenta uma metodologia híbrida denominada ATP/PL para obter

soluções variadas para problemas que consideram características de problemas reais. A ATP/PL é baseada na técnica de geração de colunas e é implementada através da iteração do Algoritmo de Treinamento Populacional (ATP), o qual é um algoritmo da família dos algoritmos genéticos, com a Programação Linear (PL). A ATP gera colunas para um problema de programação linear inteira (problema de particionamento de conjuntos), o qual é resolvido através do *software* de otimização CPLEX. Este, por sua vez, fornece informações para o ATP guiar o seu processo evolutivo e, conseqüentemente, gerar colunas ainda melhores (baixo custo e boa cobertura de tarefas). Essa metodologia se mostrou bastante robusta para resolver problemas de geração de colunas para instâncias pequenas, médias e grandes. A ATP/PL foi comparada a um algoritmo baseado na metaheurística *Simulated Annealing* (SA). A ATP/PL apresentou resultados de melhor qualidade que os produzidos pela metaheurística SA. Além disso, o método se mostrou mais robusto e muito mais rápido que o SA.

2.2.3 Programação de tripulações de trens

A programação de tripulações de trens (*train crew scheduling*) tipicamente é realizada após a programação dos trens, a partir da qual é construída uma tabela de horários a ser realizada em cada semana num certo período de tempo. Os trens operam sobre ferrovias das quais fazem parte estações e linhas férreas. Geralmente as estações estão localizadas em grandes centros urbanos ou em cidades onde as tripulações podem ser alocadas, podem descansar, ou podem trocar turnos. Quando uma tripulação passa a noite em uma estação que não a sua estação base, isto é chamado alojamento (*barracking*). As estações, além de servirem de base para as tripulações ou lugar para alojamento, são também locais onde os trens são carregados e descarregados ou são submetidos a mudanças de configuração pela adição ou remoção de vagões (*shunting*). Cada tarefa (jornada do trem) pode ser decomposta em uma seqüência de viagens, a qual é executada pela mesma tripulação sem descanso. Uma viagem é caracterizada pelo seu horário de início e estação de origem, e seu horário de término e estação de destino. As tripulações de trens geralmente executam uma jornada de trabalho, ou seja uma seqüência de tarefas (*roundtrip*), com paradas intermediárias para alojamento antes de retornar à sua estação base, onde eles têm uma longa folga antes da próxima jornada (ERNST et al., 2001).

Com a finalidade de cobrir todas as viagens, ocasionalmente é necessário para as tripulações viajarem entre estações sem conduzir um trem, operação conhecida como *paxing*. Essa estratégia de *paxing* é usualmente feita por meio de viagens em ônibus ou enviando tripulações como passageiros em outros trens. Isso conta na totalização do tempo de trabalho da tripulação por ser tratado como uma viagem normal, de forma que, para longas jornadas de *paxing*, uma tripulação precisará realizar alojamento após chegar a uma estação (ERNST et al., 2001).

O objetivo da programação de tripulações de trens é encontrar o conjunto de menor custo de jornadas que cobrem todas as viagens especificadas na tabela de horários dos trens. Custos associados a alojamento e *paxing* devem ser minimizados.

As jornadas de trabalho das tripulações precisam estar em conformidade com certas regras e restrições, tais como:

- Duração máxima de uma tarefa;
- Tempos mínimo e máximo para alojamento;
- Número máximo de tarefas numa jornada;
- Número máximo de viagens numa jornada;
- Intervalo mínimo para descanso entre duas tarefas numa jornada.

A programação de tripulações de trens apareceu apenas recentemente na literatura. O sistema TRACS II, desde 1994, foi estendido pelos seus autores para lidar com tripulações de trens (WREN, 1998). O sistema agora é usado comercialmente, e está alcançando resultados significativos (KWAN et al., 1996, 1999).

A abordagem descrita em Wren, Kwan & Kwan (2000) contempla a programação de tripulações de trens utilizando algoritmo genético.

A Busca Tabu é utilizada em Cavique, Rego & Themido (1999). Para a geração da solução inicial foi utilizado um procedimento de particionamento da tabela de horários dos trens (*run-cutting algorithm*). Com este método construtivo a tabela de horários é progressivamente coberta por *duties* até a cobertura da tabela estar completa. Durante este processo, uma viagem está coberta se existe um *duty* associado a ela e não-coberta em caso contrário. O algoritmo constrói um *duty* selecionando viagens que ainda não apresentam cobertura, a partir da tabela de horários, e cobrindo essas viagens com o novo *duty*. Este processo é repetido até que todas as viagens sejam cobertas. Na primeira estratégia uma Busca Tabu tradicional é aplicada sobre a solução inicial para minimizar o número total de *duties*. Na segunda estratégia a Busca Tabu incorpora métodos de *ejection chains* para a exploração da vizinhança. As duas estratégias tabu e o algoritmo *run-cutting* foram comparados à resolução manual. Para testar a performance do algoritmo foi utilizado um conjunto real de tabelas de horários de trens. A dimensão dos problemas-teste estava compreendida entre 528 e 718 viagens. As soluções foram comparadas em termos do número de *duties*. As soluções obtidas pelo procedimento *run-cutting* foram de qualidade inferior àquelas obtidas pela resolução manual. A estratégia tabu tradicional foi em média 1,8% melhor e a estratégia incluindo *ejection chains* foi 3,6% melhor.

2.3 Problema genérico de programação de tripulações

Nas seções anteriores foi discutida a programação de tripulações nos diversos setores de transporte. Diversos trabalhos nessa área apresentam abordagens com algoritmos fortemente dependentes das características de uma organização particular. Nesta seção, apresenta-se o Problema Genérico de Programação de Tripulações (PGPT), conhecido na literatura inglesa como *generic crew scheduling*.

O PGPT foi primeiramente discutido em Cheddad (1987) e posteriormente em Cao (1992).

Segundo Beasley & Cao (1996), esse problema é importante no sentido que expressa uma visão clara da estrutura básica dos problemas de programação de tripulações mais específicos. No topo dessa metodologia básica, restrições ou condições adicionais de algum problema de programação de tripulações particular podem ser facilmente incorporadas.

O PGPT consiste em alocar N tarefas a K ($\leq N$) *duties* (ou jornadas) onde as tarefas possuem horários fixos de início ($s_i \quad \forall\, i = 1, \ldots, N$) e término ($f_i \quad \forall\, i = 1, \ldots, N$) de forma que cada *duty* não exceda um determinado limite de tempo T. Entre duas tarefas i e j, em que o horário de término da tarefa i é igual ou inferior ao horário de início da tarefa j ($f_i \leq s_j$) e a diferença entre o horário de término da tarefa j e o horário de início da tarefa i seja igual ou inferior ao limite de tempo T ($f_j - s_i \leq T$) existe um custo de transição c_{ij} se for possível à mesma tripulação executar a tarefa i e depois a j. Assim, deve existir um tempo mínimo τ entre tarefas, ou seja, $s_j - f_i > \tau$. Dessa forma, o custo total da escala de trabalho resultante deve ser minimizado.

Conceitualmente o problema é melhor visualizado graficamente na Figura 2. Nesta figura, o eixo horizontal representa o tempo. As tarefas a serem realizadas são os vértices. Os vértices são interligados por arcos de transição. Um caminho (o qual representa um *duty*) é definido como um conjunto não-vazio de tarefas colocadas em ordem ascendente de horário de inicío. A garagem é representada duas vezes, como 0 para representar o início do caminho e como N+1 para representar o fim do caminho. Devido à dimensão tempo o grafo na figura é acíclico.

Portanto, o problema consiste em encontrar K caminhos disjuntos de 0 a N+1 na figura de forma que todas as tarefas estejam em um determinado caminho, o tempo total trabalhado num caminho não exceda um determinado limite T e o custo total dos caminhos seja minimizado.

2.3.1 Formulação matemática

O PGPT pode ser modelado como um problema de programação linear 0-1 da seguinte forma, segundo Beasley & Cao (1996):

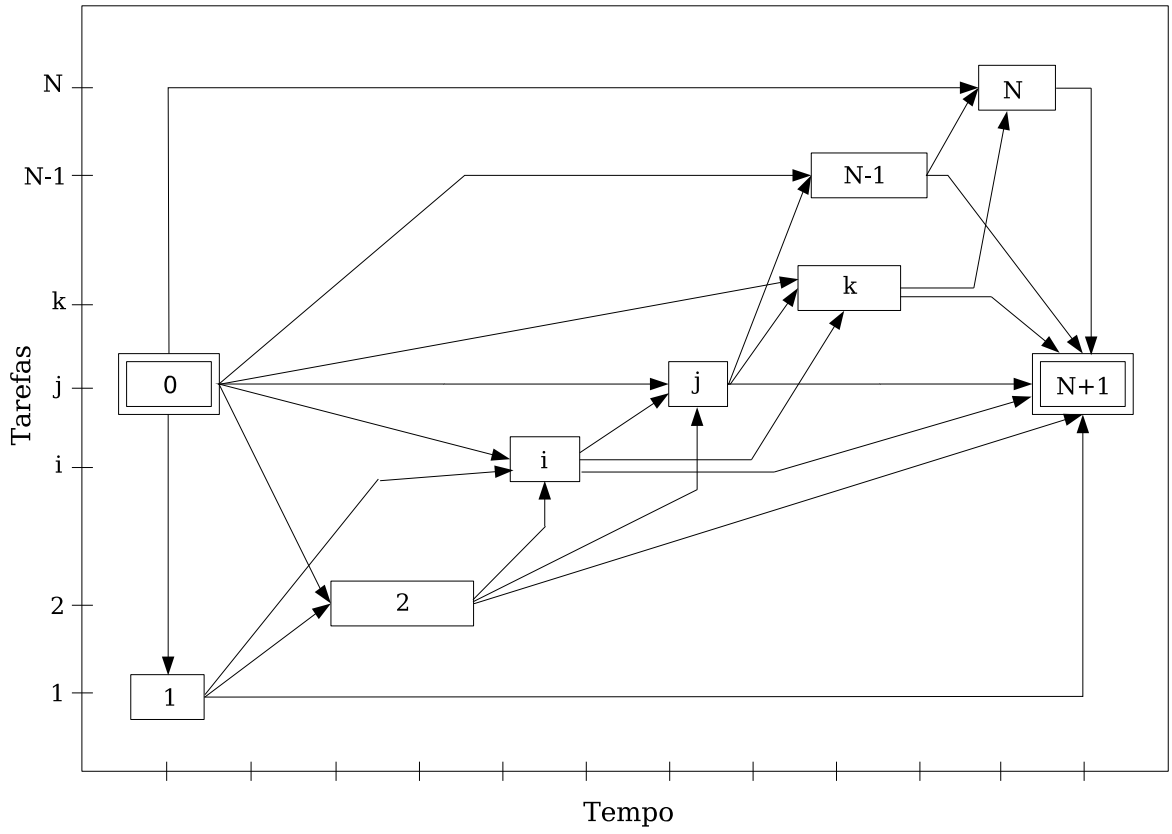

Figura 2: Visualização gráfica do PGPT

Seja:

$$x_{ij} = \begin{cases} 1 & \text{se o arco de transição(i,j) é utilizado} \\ 0 & \text{em caso contrário} \end{cases}$$

Minimizar:

$$\sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij} \tag{2.1}$$

Sujeito a:

$$\sum_{k \in N} x_{jk} = \sum_{i \in N} x_{ij} \qquad \text{j} = 1,\ldots,\text{N} \tag{2.2}$$

$$\sum_{j \in N} x_{ij} = 1 \qquad \text{i} = 1,\ldots,\text{N} \tag{2.3}$$

$$\sum_{j \in N} x_{0j} = K \tag{2.4}$$

$$\text{restrições de limite de tempo} \tag{2.5}$$

$$x_{ij} \in (0,1) \qquad \forall i,j \ \in N \tag{2.6}$$

As restrições (2.2) especificam que o número de arcos que saem de uma tarefa é igual ao número de arcos que entram nessa tarefa. As restrições (2.3) indicam que exatamente um arco sai de cada tarefa, ou seja, cada tarefa está em exatamente um caminho. A restrição (2.4) especifica que serão escolhidos no máximo K caminhos. A restrição de integralidade das variáveis x_{ij} é indicada em (2.6).

Pode ser notado pela formulação do problema que os arcos de transição da garagem para a tarefa inicial e da tarefa terminal para a garagem na formação dos caminhos, bem como os seus respectivos custos, não são levados em consideração na função objetivo.

2.4 Método VNS-RTL

Souza et al. (2004) propuseram o método VNS-RTL para abordar o Problema de Programação Diária de Tripulações de uma empresa do Sistema de Tranporte Público que opera na cidade de Belo Horizonte. Este método é baseado no método VNS e utiliza o método VND como método de busca local. Este trabalho será descrito com maiores detalhes já que será usado para comparação com os métodos propostos por este trabalho.

O Método de Pesquisa em Vizinhança Variável, conhecido como VNS (*Variable Neighborhood Search*), proposto por Mladenovic & Hansen (1997), é um método de busca local que consiste em explorar o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança. A idéia do método é a de explorar vizinhanças gradativamente mais "distantes" da solução corrente, focando a busca em torno de uma nova solução se e somente se um movimento de melhora é realizado. O método VNS também inclui um procedimento de busca local a ser aplicado sobre o vizinho da solução corrente. O pseudo-código desse algoritmo é apresentado pela Figura 3.

Algoritmo `VNS`

1: Entrada: $f(.), N(.), r, s_0$
2: $s \leftarrow s_0$;
3: **enquanto** Critério de parada não satisfeito **faça**
4: $k \leftarrow 1$; {Tipo de estrutura de vizinhança corrente}
5: **enquanto** $k \leq r$ **faça**
6: Gere um vizinho qualquer $s' \in N^{(k)}(s)$;
7: $s'' \leftarrow BuscaLocal(s')$;
8: **se** $f(s'') < f(s)$ **então**
9: $s \leftarrow s''$;
10: $k \leftarrow 1$;
11: **senão**
12: $k \leftarrow k + 1$;
13: **fim se**
14: **fim enquanto**
15: **fim enquanto**
16: Retorne s;

Figura 3: Método de Pesquisa em Vizinhança Variável

O Método de Descida em Vizinhança Variável, conhecido como VND (*Variable Neighborhhod Descent*), proposto em Mladenovic & Hansen (1997), é um procedimento de busca local. Este consiste nas linhas 4 a 14 da Figura 3, excetuando-se a linha 7 e substituindo-se a linha 6 por "Encontre o melhor vizinho $s' \in N^{(k)}(s)$". Desta forma, o método pára quando for atingida a última estrutura de vizinhança e nenhuma melhora na solução corrente for possível. Ao contrário do método VND, o método VNS tem condições de prosseguir a busca quando essa última situação ocorre, uma vez que retorna-se à primeira

estrutura de vizinhança e seleciona-se um outro vizinho qualquer até que uma determinada condição de parada seja satisfeita.

O VNS-RTL utiliza três estruturas de vizinhança, $N^R(s)$ (Realocação), $N^T(s)$ (Troca) e $N^L(s)$ (*Link*), explorando cada vizinhança nesta ordem. A Figura 4 ilustra cada uma das estruturas mencionadas. Nesta figura, $A, \ldots, G$ representam tarefas a serem executadas nas jornadas i e j.

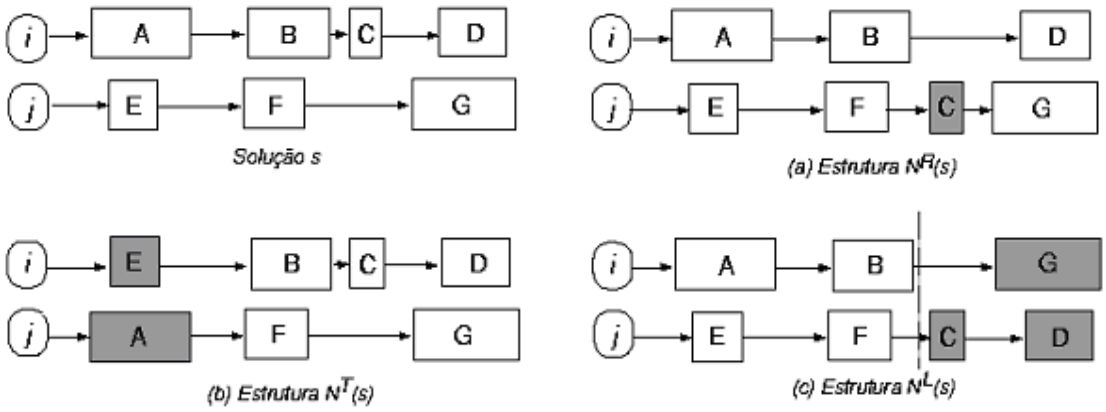

Figura 4: Estruturas de Vizinhança do método VNS-RTL

A estrutura Realocação baseia-se no movimento de realocação, consistindo em realocar uma tarefa de uma jornada para outra. Em (a), mostra-se que a tarefa C anteriormente pertencente à jornada i é realocada à jornada j. O conjunto de todos os vizinhos de s gerados através de movimentos de realocação define a estrutura de vizinhança $N^R(s)$.

A estrutura Troca é definida a partir do movimento de troca, que consiste na permuta de tarefas entre duas jornadas distintas. Em (b), mostra-se que as tarefas A e E são trocadas entre as jornadas i e j. O conjunto de todos os vizinhos de s gerados através de movimentos de troca define a estrutura de vizinhança $N^T(s)$.

Já a estrutura *Link* é baseada no movimento *Link* que consiste na troca de um conjunto de tarefas entre duas jornadas distintas. Conforme ilustrado em (c), a partir de uma solução s define-se um ponto de corte em uma das jornadas, horário a partir do qual os blocos de tarefas são trocados. O conjunto de todos os vizinhos de s gerados a partir de movimentos do tipo *Link* define a estrutura de vizinhança $N^L(s)$.

Dada a elevada complexidade para avaliação de uma solução no movimento *Link*, a cada iteração do método VND, ordenam-se inicialmente as jornadas de acordo com suas durações. A seguir consideram-se *MaiorJorn*% das jornadas com maior duração e *MenorJorn*% das jornadas de menor duração. Então, a partir de um ponto de corte definido para cada umas das jornadas de maior duração, aplica-se o movimento *Link*

considerando-se que um bloco de tarefas de uma dessas jornadas seja permutado com um bloco de tarefas iniciado a partir do mesmo ponto de corte em uma das jornadas de menor duração.

Para minimizar o esforço computacional na exploração de soluções nas três estruturas de vizinhança, evita-se a análise completa de uma dada vizinhança em cada iteração do método VND, interrompendo-se a busca na vizinhança corrente sempre que ocorrer uma solução de melhora.

A solução inicial é gerada tomando-se como base a filosofia adotada pela empresa na construção manual de suas jornadas. Esta metodologia baseia-se na repartição dos blocos dos veículos. Cada bloco de um veículo contém um conjunto de tarefas a serem executadas pelo veículo no dia. Este bloco é dividido seguindo dois critérios: sua duração e o fato de o veículo retornar ou não à garagem mais de uma vez durante a sua jornada diária. Caso o veículo não retorne à garagem mais de uma vez, a divisão do bloco é feita de acordo com a sua duração. Nesta situação, o bloco é dividido ao meio caso cada metade não ultrapasse 9:10 horas de trabalho, que é o tempo máximo de trabalho permitido por lei, incluindo as 2:00 horas extras. Caso contrário, o bloco é dividido em três partes, sendo as duas primeiras de no máximo 7:10 horas e a última com as tarefas remanescentes. Cada parte de um bloco é alocada a uma jornada diferente.

Caso o veículo retorne à garagem durante a sua jornada, a repartição do bloco também se dá conforme a sua duração, porém a divisão é feita no intervalo de tempo em que a diferença entre o final da última tarefa e o início da primeira tarefa, excluindo o tempo em que o veículo ficou na garagem, for inferior a 8:40 horas. O tempo em que o veículo permanece na garagem é excluído porque o tripulante que receber essa jornada fará dupla pegada e, portanto, não necessitará dos 30 minutos de intervalo para repouso e/ou alimentação.

Segundo os autores, uma das vantagens de se gerar a solução inicial seguindo a filosofia da empresa é que esta não contém sobreposição de tarefas nem trocas de pontos ou linhas proibidas. Todavia, pode ocorrer excesso na duração das jornadas diárias de trabalho.

Os autores testaram o método considerando dados relativos a um dia útil da empresa, a qual é responsável por 11 linhas de ônibus, com frota empenhada de 101 veículos. O algoritmo foi executado 20 vezes, cada qual partindo de uma semente diferente de números aleatórios. A Tabela 1 apresenta as características da melhor solução encontrada.

No seção a seguir, é descrita a heurística Busca Tabu que será utilizada na solução do problema abordado neste trabalho.

Tabela 1: Melhor solução obtida pelo método VNS-RTL

Melhor solução	
Valor	1.188.980
Número de Jornadas	219
Hora-extra (hh:mm)	87:13
Trocas de veículos	28
Trocas de linhas	13
Trocas de pontos	11

2.5 Busca Tabu

O método de Busca Tabu (BT), proposto independentemente por Glover (1986) e Hansen (1986), é um procedimento iterativo para a solução de problemas de otimização combinatória que aceita movimentos de piora para tentar escapar de ótimos locais distantes de um ótimo global.

Começando com uma solução inicial s_0, o método explora a cada iteração, um subconjunto V da vizinhança $N(s)$ da solução corrente s. Considerando o problema de minimização de uma função $f(.)$, o membro s' de V com menor valor nessa região torna-se a nova solução corrente mesmo que s' seja pior que s, isto é que $f(s') > f(s)$. No entanto, a melhor solução gerada até o momento, s^*, é sempre armazenada.

O critério de escolha da nova semente é utilizado para tentar escapar de um mínimo local. Esta estratégia, entretanto, pode fazer com que o algoritmo cicle, isto é, que retorne a uma solução já gerada anteriormente.

De forma a evitar que isto ocorra, existe um mecanismo chamado *mémoria de curto prazo* ou lista tabu. O objetivo dessa lista é tentar evitar movimentos que levem à regiões já visitadas do espaço de soluções, o que usualmente é alcançado pela proibição dos últimos movimentos realizados. Esses movimentos são armazenados nesta lista e permanecem proibidos (com *status* tabu), por um dado número de iterações, chamado *tabu tenure*. Assim, na exploração do subconjunto V da vizinhança $N(s)$ da solução corrente s, ficam excluídos da busca os vizinhos s' que são obtidos de s por movimentos m que constam na lista tabu. O uso da lista tabu pode ser muito restritivo para a busca no espaço de soluções, assim, para não desconsiderar soluções de alta qualidade, movimentos com *status* tabu podem ser aceitos se a nova solução produzida satisfaz um *critério de aspiração*. Este é um mecanismo que retira, sob certas circunstâncias, o *status* tabu de um movimento. Deste modo, para cada possível valor v da função objetivo existe um nível de aspiração $A(v)$: uma solução s' em V pode ser considerada se $f(s') \leq A(f(s))$, mesmo que o movimento m esteja na lista tabu. A função de aspiração A é tal que, para cada valor v da função objetivo, retorna outro valor $A(v)$, que representa o valor que o

algoritmo aspira ao chegar de v. Um critério de aspiração comumente adotado é o critério de aspiração por objetivo. Por este critério, uma solução s', ainda que seja gerada através de um movimento tabu, é aceita se $f(s')$ for melhor que $f(s^*)$ (no caso de minimização, $f(s') < f(s^*)$).

Duas regras são normalmente utilizadas de forma a interromper o procedimento. Pela primeira, a execução é interrompida quando atinge-se um certo número de iterações sem melhora no valor da melhor solução. Pela segunda, quando o valor da melhor solução chega a um limite inferior conhecido (ou próximo dele). Esse segundo critério evita a execução desnecessária do algoritmo quando uma solução ótima é encontrada ou quando uma solução é julgada suficientemente boa.

Apresenta-se pela Figura 5 o algoritmo de Busca Tabu clássico.

Algoritmo `BT`

1: Entrada: $f(.), N(.), A(.), |V|, f_{min}, |T|, BTmax, s$
2: $s^* \leftarrow s$;
3: $Iter \leftarrow 0$;
4: $MelhorIter \leftarrow 0$;
5: $T \leftarrow \oslash$;
6: Inicialize a função de aspiração A;
7: **enquanto** Critério de parada não satisfeito **faça**
8: $\quad Iter \leftarrow Iter + 1$;
9: $\quad$ Seja $s' \leftarrow s \oplus m$ o melhor elemento de $V \subset N(s)$ tal que o movimento m não seja tabu ($m \notin T$) ou $f(s') < A(f(s))$;
10: $\quad$ Atualize a lista tabu;
11: $\quad s \leftarrow s'$;
12: $\quad$ **se** $f(s) < f(s^*)$ **então**
13: $\quad\quad s^* \leftarrow s$;
14: $\quad\quad MelhorIter \leftarrow Iter$;
15: $\quad$ **fim se**
16: $\quad$ Atualize a função de aspiração A;
17: **fim enquanto**
18: $s \leftarrow s^*$;
19: Retorne s;

Figura 5: Algoritmo de Busca Tabu

Os parâmetros principais do método de Busca Tabu são a cardinalidade $|T|$ da lista tabu, a função de aspiração A, a cardinalidade do conjunto $|V|$ de soluções vizinhas testadas em cada iteração, $BTmax$ que é o número de iterações sem melhora no valor da melhor solução e f_{min}, o valor mínimo conhecido da função f, informação essa que em alguns casos está disponível.

2.6 Módulos adicionais na Busca Tabu

Descreve-se, a seguir, módulos que têm sidos incorporados, com mais freqüência, à Busca Tabu da Figura 5.

2.6.1 Estratégias de intensificação

É comum em métodos de Busca Tabu incluir estratégias de intensificação, as quais tem por objetivo efetuar buscas mais refinadas em determinadas regiões consideradas promissoras do espaço de soluções. Abaixo segue uma breve descrição de algumas das principais estratégias de intensificação descritas por Glover & Laguna (1997).

2.6.1.1 Reinício da BT com soluções elite

Nessa estratégia mantém-se armazenada uma lista das melhores soluções distintas encontradas até o momento pelo algoritmo (soluções elite) e estas são utilizadas como base para o processo de reinício do método de BT. A lista de soluções elite geralmente apresenta um tamanho limitado de forma que faz-se necessário um critério para a atualização da mesma.

2.6.1.2 Freqüência das soluções elite

Nessa estratégia as melhores soluções encontradas são investigadas para a determinação da freqüência de determinados atributos. Usualmente atribui-se um peso para a freqüência de um atributo de acordo com a qualidade das soluções nas quais este foi encontrado. Estratégias como esta torna possível determinar qual o percentual de bons atributos das soluções elite encontradas deve ser incluído na solução atualmente pesquisada. Também é possível determinar subconjuntos no quais os atributos são mantidos juntos de forma a guiar o processo de busca para que não sejam feitas determinadas escolhas que deteriorem a qualidade da solução quando comparado a outras escolhas.

2.6.1.3 Memória e intensificação

Nessa estratégia incorpora-se atributos das soluções elite e estimula-se atributos destas soluções a fazerem parte da solução corrente. Nesse caso, são considerados livres no procedimento de busca local apenas os atributos não associados às soluções elite, permanecendo os demais atributos fixos. Um critério de aspiração torna possível atributos fixos serem considerados livres, dando a essa abordagem maior flexibilidade.

2.6.2 Estratégias de diversificação

Métodos de Busca Tabu podem incluir, também, estratégias de diversificação. O objetivo dessas estratégias, que podem fazer uso de *memórias de longo prazo*, é redirecionar a pesquisa para regiões ainda não suficientemente exploradas do espaço de soluções. Apresenta-se, a seguir, uma breve descrição de algumas das principais estratégias de diversificação descritas por Glover & Laguna (1997).

2.6.2.1 Freqüência das soluções não-elite

Nessa abordagem mantém-se a informação da freqüência de atributos de soluções não-elite. Com base nessa análise desistimula-se, através de penalidades, a incorporação de atributos que se apresentam freqüentes nas soluções elite. Caso um atributo não tenha aparecido ou raramente apresentou-se em soluções geradas anteriormente, estimula-se a incorporação do mesmo na solução corrente.

2.6.2.2 Busca Tabu Reativa

A BT reativa, proposta originalmente por Battiti & Tecchiolli (1992), tem por objetivo alcançar uma boa combinação entre intensificação e diversificação, propondo uma maneira alternativa para detectar ciclagem. A detecção de ciclos é feita checando-se a repetição de determinados atributos de soluções os quais permitem diferenciá-las tanto quanto possível, ou seja, poucas soluções devem possuir os mesmos atributos. Para esse propósito utiliza-se uma estrutura de *hashing* para a otimização da consulta. Nesse método, ciclos potenciais são detectados pela comparação da freqüência de repetição com um dado número limiar estimado. Inicialmente a lista tabu possui um tamanho razoavelmente pequeno (podendo ser igual a 0). Caso seja detectado um ciclo, o tamanho da lista é ajustado para o comprimento do mesmo. Em caso contrário, decrementa-se exponencialmente o tamanho da lista tabu se o mesmo ultrapassar um determinado valor. Um procedimento de diversificação é ativado caso o número de ciclos seja excessivamente grande.

2.6.2.3 Abordagens por *Ejection Chains*

Os métodos baseados em *ejection chains* constituem uma combinação implícita entre intensificação e diversificação (CAVIQUE; REGO; THEMIDO, 1999). Nestes, avalia-se a vizinhança de uma solução por transformações sucessivas de uma estrutura, que normalmente não representa uma solução completa ou factível, mas é utilizada como referência para a avaliação de soluções *teste* em cada passo da *ejection chain*. A tranformação de uma estrutura em outra consiste em substituir alguns atributos da estrutura corrente por novos atributos, forçando outros a deixarem a mesma, o que define um *ejection move*.

Cada transformação da estrutura define um *nível* da *ejection chain* na qual uma solução teste pode ser obtida por um movimento *teste* apropriado. Em cada nível n da cadeia (*chain*) existe um movimento composto produzido pelo efeito acumulativo de $n-1$ *ejection moves* anteriores seguido pelo movimento teste no nível n. A pesquisa em vizinhança geralmente consiste de construir N níveis de uma *ejection chain* a partir da solução corrente, avaliando-se as soluções teste em cada nível. O nível l' no qual a melhor solução teste é encontrada determina o movimento composto usado para transformar a solução corrente em outra soluçao.

2.6.3 Reconexão por Caminhos

A técnica de reconexão por caminhos (*Path Relinking*) foi prosposta por Glover (1996) como uma estratégia de intensificação, explorando trajetórias que conectavam soluções elite obtidas por Busca Tabu ou *Scatter Search* (GLOVER; LAGUNA, 1997). A busca por soluções de melhor qualidade consiste em gerar e explorar caminhos no espaço de soluções partindo de uma ou mais soluções elite e levando a outras soluções elite. Para tal finalidade, seleciona-se movimentos que introduzem atributos das soluções guia na solução corrente. Deste modo, a reconexão por caminhos pode ser vista como uma estratégia que objetiva incorporar atributos de soluções de boa qualidade, favorecendo a seleção de movimentos que as contenham.

A reconexão por caminhos pode ser aplicada segundo duas estratégias básicas (ROSSETI, 2003):

- reconexão por caminhos aplicada como uma estratégia de pós-otimização entre todos os pares de soluções elite;
- reconexão por caminhos aplicada como uma estratégia de intensificação a cada ótimo local obtido após a fase de busca local.

A aplicação da técnica de reconexão por caminhos como um procedimento de intensificação a cada ótimo local é mais eficaz do que empregá-la como um procedimento de pós-otimização segundo Rosseti (2003). Neste caso, a reconexão por caminhos é aplicada a pares (s_1, s_2) de soluções, onde s_1 é a solução corrente obtida após o procedimento de busca local e s_2 é uma solução selecionada aleatoriamente de um conjunto formado por um número limitado, $TamConjElite$, de soluções elite encontradas durante a Busca Tabu. Este conjunto está, inicialmente, vazio. Cada solução obtida ao final de uma busca local é considerada como uma candidata a ser inserida no conjunto elite, se ela é melhor que a solução de pior qualidade deste conjunto e apresenta um percentual mínimo de diferença em relação a cada solução do conjunto elite ($PercDif$). Se o conjunto estiver vazio a solução é simplesmente inserida no conjunto. Se o conjunto elite já possui

$TamConjElite$ soluções e a solução corrente é candidata a ser inserida neste conjunto, então esta substitui a solução de pior qualidade.

O algoritmo inicia computando a diferença simétrica $\Delta(s_1, s_2)$ entre s_1 e s_2, resultando no conjunto de movimentos que deve ser aplicado a uma delas (solução inicial) para alcançar a outra (a solução guia). A partir da solução inicial, o melhor movimento ainda não executado de $\Delta(s_1, s_2)$ é aplicado à solução corrente, até que a solução guia seja atingida. A melhor solução encontrada ao longo desta trajetória é considerada como candidata à inserção no conjunto elite e a melhor solução globalmente já encontrada é atualizada. Diversas alternativas têm sido consideradas e combinadas em implementações recentes (ROSSETI, 2003):

- não aplicar reconexão por caminhos a cada iteração, mas sim periodicamente, para não onerar o tempo final do algoritmo;
- explorar duas trajetórias potencialmente diferentes, usando s_1 como solução inicial e depois s_2 no mesmo papel (*forward, backward*);
- explorar apenas uma trajetória, usando s_1 ou s_2 como solução inicial; e
- não percorrer a trajetória completa de s_1 até s_2, mas sim apenas parte dela (reconexão por caminhos truncada).

Para a escolha da alternativa mais apropriada deve-se ter um compromisso entre tempo de processamento e qualidade da solução. Em Ribeiro, Uchoa & Werneck (2002) foi observado que a exploração de duas trajetórias potencialmente diferentes para cada par (s_1, s_2) consome, aproximadamente, o dobro do tempo de processamento necessário para explorar apenas uma delas, com um ganho marginal muito pequeno em termos de qualidade de solução. Também foi observado pelos autores que, se apenas uma trajetória deve ser investigada, melhores soluções tendem a ser obtidas quando a reconexão por caminhos usa como solução inicial a melhor solução dentre s_1 e s_2. Como a vizinhança da solução inicial é explorada com muito mais profundidade do que aquela da solução guia, usar como solução inicial a melhor dentre s_1 e s_2 oferece mais chances ao algoritmo de investigar mais detalhadamente a vizinhança da solução mais promissora. Pela mesma razão, as melhores soluções são normalmente encontradas próximas da solução inicial, permitindo que o procedimento de reconexão por caminhos seja interrompido após algumas iterações, antes da solução guia ser alcançada.

Apresenta-se pela Figura 6 o pseudo-código contendo os principais blocos do algoritmo de Busca Tabu com intensificação por reconexão por caminhos para um problema de minimização.

Algoritmo BTPR

1: Entrada: $f(.), N(.), A(.), |V|, f_{min}, |T|, BTmax, s, IterAtivRC, IteracoesRC$
2: $s^* \leftarrow s$;
3: $Iter \leftarrow 0$;
4: $MelhorIter \leftarrow 0$;
5: $T \leftarrow \oslash$;
6: $ConjuntoElite \leftarrow \oslash$;
7: $IterSemMelhora \leftarrow 0$;
8: Inicialize a função de aspiração A;
9: **enquanto** Critério de parada não satisfeito **faça**
10: $Iter \leftarrow Iter + 1$;
11: Seja $s' \leftarrow s \oplus m$ o melhor elemento de $V \subset N(s)$ tal que o movimento m não seja tabu ($m \notin T$) ou $f(s') < A(f(s))$;
12: Atualize a lista tabu;
13: $s \leftarrow s'$;
14: **se** s deve pertencer ao conjunto de soluções elite **então**
15: Inserir s no *ConjuntoElite*;
16: **fim se**
17: **se** ($IterSemMelhora$ **mod** $IterAtivRC < IteracoesRC$) **and** ($Iter \geq IterAtivRC$) **então**
18: Escolher aleatoriamente $g \in ConjuntoElite$ com probabilidade uniforme;
19: Atribuir a g' a melhor solução obtida aplicando reconexão por caminhos ao par (s, g);
20: $s \leftarrow g'$;
21: **fim se**
22: **se** g' deve pertencer ao conjunto de soluções elite **então**
23: Inserir g' no *ConjuntoElite*;
24: **fim se**
25: **se** $f(s) < f(s^*)$ **então**
26: $s^* \leftarrow s$;
27: $MelhorIter \leftarrow Iter$;
28: $IterSemMelhora \leftarrow 0$;
29: **senão**
30: $IterSemMelhora \leftarrow IterSemMelhora + 1$;
31: **fim se**
32: **fim enquanto**
33: $s \leftarrow s^*$;
34: Retorne s;

Figura 6: Algoritmo de Busca Tabu com Reconexão por Caminhos

A Figura 7 representa o pseudo-código do procedimento de reconexão por caminhos para um problema de minimização. O algoritmo de reconexão por caminhos unidirecional inicia determinando o conjunto de movimentos $\Delta(s, g)$ que será aplicado a s (solução inicial) até chegar a g (solução guia) (linha 3). Cada iteração do procedimento de reconexão por caminhos unidirecional possui os quatro passos seguintes:

- aplicar à solução corrente $\bar{g}$ o melhor movimento do conjunto de movimentos (linha 5), obtendo a solução g'';
- excluir o melhor movimento do conjunto de movimentos ainda possível (linha 6);
- atualizar a solução corrente (linha 7); e
- testar se a solução corrente, $\bar{g}$, é melhor que a melhor solução, g', encontrada ao longo da trajetória aplicada a s para chegar a g. Em caso afirmativo, atribui-se $\bar{g}$ a g' (linha 9). No início do método de reconexão por caminhos, atribui-se a g' a melhor solução dentre s e g (linha 2).

Quando a solução corrente chega a g, o algoritmo de reconexão por caminhos pára (linha 4) e retorna a solução g' (linha 12).

Procedimento `Reconexão-Caminhos`

1: $\bar{g} \leftarrow s$;
2: Atribuir a g' a melhor solução entre s e g;
3: Calcular o conjunto de movimentos possíveis $\Delta(s, g)$;
4: **enquanto** $|\Delta(s, g)| \neq 0$ **faça**
5: Atribuir a g'' a melhor solução obtida aplicando o melhor movimento de $\Delta(s, g)$ a $\bar{g}$;
6: Excluir de $\Delta(s, g)$ este movimento;
7: $\bar{g} \leftarrow g''$;
8: **se** $f(\bar{g}) < f(g')$ **então**
9: $g' \leftarrow \bar{g}$;
10: **fim se**
11: **fim enquanto**
12: Retorne g';

Figura 7: Procedimento de Reconexão por Caminhos

2.6.4 Relaxação Adaptativa

A Relaxação Adaptativa (também conhecida como Oscilação Estratégica) está intimamente ligada às origens da Busca Tabu (GLOVER; LAGUNA, 1997). Esta técnica provê um meio de se alcançar uma combinação entre intensificação e diversificação. Ela consiste em orientar movimentos em relação a um nível crítico, que pode ser identificado por um

estágio de construção ou um intervalo escolhido de valores para uma função. Tal nível crítico (ou fronteira de oscilação) freqüentemente representa um ponto onde o método normalmente iria ser interrompido. Em vez de parar quando esta fronteira é alcançada, as regras para selecionar movimentos são modificadas para permitir que a região definida pelo nível crítico seja atravessada. Esta abordagem então continua até uma dada profundidade além da fronteira de oscilação e volta novamente. O limite da oscilação é novamente alcançado e atravessado, desta vez na direção oposta, e assim por diante (Figura 8).

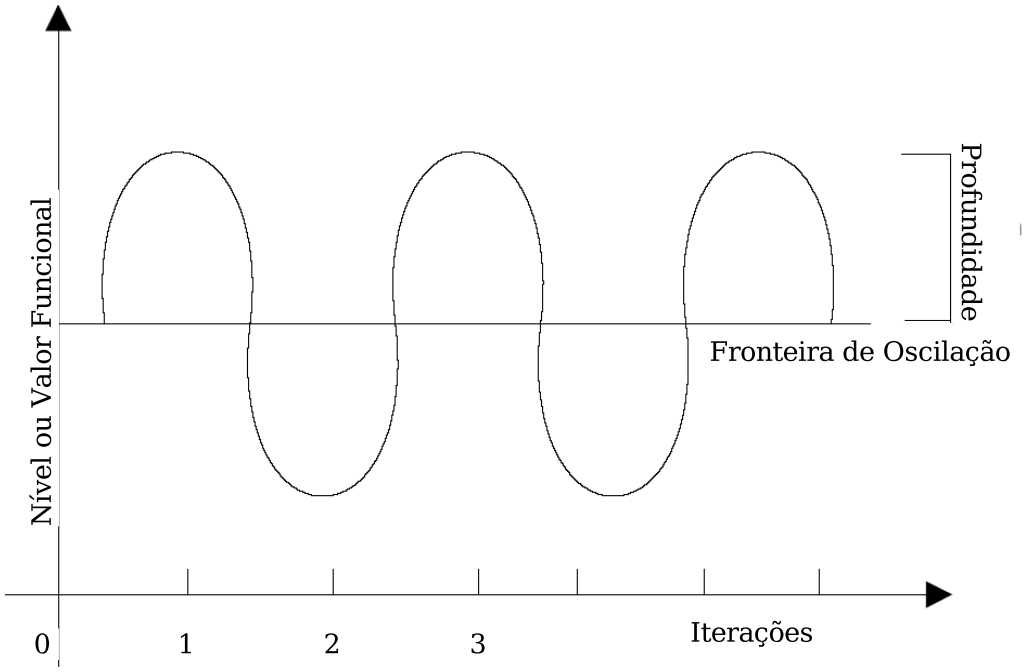

Figura 8: Oscilação Estratégica

O processo de repetidamente abordar e atravessar o nível crítico a partir de diferentes direções cria um comportamento oscilatório, o qual dá o nome ao método. O controle sobre esse comportamento é estabelecido por meio da geração de avaliações e regras modificadas para os movimentos, dependendo da região navegada e da direção da busca. A possibilidade de percorrer uma trajetória já visitada é evitada por mecanismos tabu padrões, tais como aqueles estabelecidos pelas funções da memória de curto prazo.

(SCHAERF, 1996) apresenta um mecanismo de relaxação adaptativa onde os pesos, para cada fonte de inviabilidade da função de avaliação, são ajustados dinamicamente como proposto por Gendreau, Hertz & Laporte (1994). Para cada fonte de inviabilidade i o peso W_i é multiplicado por um fator α_i que varia de acordo com o seguinte esquema:

1. No início da busca $\alpha_i \leftarrow 1$.

2. A cada k movimentos:

   - se todas as k soluções visitadas são factíveis em relação à inviabilidade i então $\alpha_i \leftarrow \alpha_i/\gamma$;
   - se todas as k soluções visitadas são infactíveis em relação à inviabilidade i então $\alpha_i \leftarrow \alpha_i \times \gamma$;
   - se algumas soluções são factíveis e algumas outras são infactíveis então α_i permanece inalterado.

O parâmetro γ é randomicamente selecionado, a cada vez, no intervalo $[1, 8, 2, 2]$. Em Gendreau, Hertz & Laporte (1994), γ é deterministicamente fixado no valor 2. Schaerf (1996) optou por randomizar tal valor para evitar que escolhas determinísticas pudessem guiar a busca.

Cada valor de α_i é limitado por duas constantes $\alpha_{i,min}$ e $\alpha_{i,max}$. Isso implica que se α_i assumir um valor superior a $\alpha_{i,max}$, então ele recebe o valor $\alpha_{i,max}$. De maneira semelhante, se α_i assumir um valor inferior a $\alpha_{i,min}$, ele recebe o valor $\alpha_{i,min}$. A limitação do valor de α_i tem como objetivo não perder o referencial de avaliação das soluções, fato que ocorreria após uma longa seqüência de soluções infactíveis ou factíveis, em decorrência de valores muito altos ou muito baixos, respectivamente, para esse parâmetro.

3 O Problema de Programação de Tripulações de Ônibus Urbano

Apresenta-se neste capítulo a descrição detalhada do Problema de Programação de Tripulações abordado, relativo a uma empresa de transporte público da cidade de Belo Horizonte-MG, com regras trabalhistas relativas ao período 2002/2003.

Como dito anteriormente, no transporte público a Programação de Tripulações é usualmente feita a partir da Programação de Veículos. Nesta, as viagens são reunidas em *blocos*. Um bloco apresenta a seqüência de viagens que um determinado veículo deve realizar em um dia, começando e terminando na garagem. Cada bloco apresenta as *Oportunidades de Troca* (OTs). A OT é um intervalo de tempo suficiente para que ocorra a troca de tripulações. A cada oportunidade de troca está associado um *Horário de Troca* e um *Ponto de Troca*, os quais representam respectivamente a hora e um local onde poderá haver uma troca de tripulação.

A partir do bloco de um veículo são criadas as *Tarefas*. Cada tarefa é um conjunto de viagens reunidas de maneira que haja apenas duas OTs: uma no início e outra no final da tarefa. Assim, durante sua realização não é possível que haja troca de tripulação.

A Figura 9 ilustra um bloco de um veículo particionado em tarefas. Supõe-se, neste exemplo, que o tempo mínimo para troca de tripulações é de 6 minutos. A coluna *Viagem* indica o número de viagens realizadas pelo veículo até então. As colunas *Ponto de Partida* e *Horário de Partida* indicam respectivamente o local e o horário de saída do veículo. As colunas *Ponto de Chegada* e *Horário de Chegada* representam respectivamente o local e o horário de chegada do veículo. O ponto 0 representa a garagem. A coluna *Viagem Vazia* representa a quantidade de tempo em minutos em que o veículo não está transportando passageiros e a coluna *Tempo de Terminal*, o tempo em que o veículo permanece no terminal aguardando a próxima viagem programada. A coluna *Linha* representa a linha à qual o veículo está alocado.

Conforme se observa nessa figura, a tarefa 5 reune as viagens 05 e 06. Isto ocorre porque o intervalo de 5 minutos entre elas não é suficiente para a troca de tripulações, operação que exige 6 minutos. Portanto, esta tarefa acumula uma folga de 5 minutos, que é o intervalo entre essas viagens.

Viagem	Ponto de Partida	Horário de Partida	Ponto de Chegada	Horário de Chegada	Viagem Vazia	Tempo de Terminal	Linha	
00	0	04:50	1	05:00	10	00	101	**Tarefa 1**
01	1	05:00	1	05:45	00	15	101	
02	1	06:00	1	06:45	00	06	101	**Tarefa 2**
03	1	06:51	1	07:36	00	06	101	**Tarefa 3**
04	1	07:42	1	08:27	00	06	101	**Tarefa 4**
05	1	08:33	1	09:15	00	05	101	**Tarefa 5**
06	1	09:20	1	10:00	00	06	101	
07	1	10:05	1	10:49	00	06	101	**Tarefa 6**
08	1	10:55	1	11:39	00	06	101	**Tarefa 7**
09	1	11:45	1	12:29	00	06	101	**Tarefa 8**
10	1	12:35	1	13:19	00	06	101	**Tarefa 9**
11	1	13:25	1	14:09	00	06	101	**Tarefa 10**
12	1	14:15	1	14:59	00	06	101	**Tarefa 11**
13	1	15:05	1	15:50	00	14	101	**Tarefa 12**
14	1	16:04	1	16:49	00	06	101	**Tarefa 13**
15	1	16:55	1	17:40	00	06	101	**Tarefa 14**
16	1	17:46	1	18:31	00	06	101	**Tarefa 15**
17	1	18:37	1	19:22	00	00	101	**Tarefa 16**
00	1	19:22	0	19:32	10	00	101	

Figura 9: Bloco de um veículo particionado em tarefas

A programação de uma tripulação é formada por um conjunto de tarefas, chamado de *Jornada*. As jornadas são divididas em dois tipos: *Pegada simples* ou *Dupla pegada*. No primeiro tipo as tarefas são realizadas de uma única vez e os intervalos entre elas são menores que duas horas. Caso ocorra um intervalo maior ou igual a duas horas a jornada é classificada como dupla pegada. Este intervalo não é contabilizado na remuneração da tripulação. A duração normal de uma jornada, seja ela de *Pegada simples* ou de *Dupla pegada* é de 6:40 horas de efetivo trabalho. No caso de *Pegada simples*, o tripulante tem direito a um intervalo de 30 minutos para descanso e/ou alimentação. Assim, a duração de sua jornada estende-se a 7:10 horas.

Ao se reunir as tarefas formando as jornadas deve-se levar em conta as seguintes restrições operacionais e trabalhistas:

a) Uma tripulação não pode realizar mais de uma tarefa ao mesmo tempo (*Sobreposição de Tarefas*);

b) Uma tripulação não pode realizar duas tarefas consecutivas na qual o ponto de troca do final da primeira seja diferente do ponto de troca do início da segunda, e o intervalo entre elas seja menor do que duas horas (*Troca de Pontos Proibida*);

c) Por determinações trabalhistas uma tripulação não pode exercer mais que 2 (duas) horas de atividades diárias além da duração normal de sua jornada (*Horas Excedentes*);

d) Durante a realização de uma jornada de trabalho a tripulação tem direito a uma folga de 30 minutos destinada à refeição. Por acordo feito com os trabalhadores, esta folga pode ser dividida e distribuída no decorrer do dia, conquanto que uma das frações seja de pelo menos 15 minutos. O método desenvolvido contempla automaticamente esse quesito, sendo que este não é contabilizado no cálculo da função de avaliação (*Folga Acumulada Deficiente*);

e) O tempo mínimo entre jornadas de trabalho é de 11:00 horas. A jornada da tripulação que não atingir esse valor será penalizada pelo intervalo de tempo necessário para se atingir o tempo mínimo (*Tempo entre Jornadas*);

f) Uma tripulação não pode realizar duas tarefas consecutivas na qual o grupo da linha final da primeira seja diferente do grupo da linha inicial da segunda, isto é, um veículo que faz um certo itinerário não pode fazer logo a seguir outro itinerário que não esteja em uma mesma região definida pela empresa (*Troca de Linhas Proibida*);

g) O número de tripulações que têm jornadas do tipo dupla pegada não pode superar um certo número (*Número de duplas pegadas*);

h) O número de horas que superam a duração normal de uma jornada é considerado hora-extra (*Horas Extras*);

i) O número de trocas de pontos nos intervalos que caracterizam jornadas do tipo dupla pegada deve ser minimizado. Isto é, evitar sempre que possível que a última tarefa da primeira pegada de uma tripulação ocorra em um ponto diferente da primeira tarefa da segunda pegada (*Troca de Pontos Permitida*);

j) O número de trocas de veículos deve ser minimizado. Isto é, reduzir o número de vezes em que uma tripulação realiza duas tarefas consecutivas em veículos diferentes (*Troca de Veículos*);

k) O número total de jornadas de trabalho deve ser minimizado (*Número de jornadas*);

l) Reduzir, tanto quanto possível a ociosidade nas jornadas (*Ociosidade*);

m) O número de vezes que uma tripulação troca de linha dentro de um mesmo grupo deve ser minimizado (*Troca de Linhas Permitida*).

As sete primeiras restrições ((a)...(g)) são consideradas essenciais, ou seja, se não forem atendidas a solução obtida é infactível. As demais restrições ((h)...(m)) são consideradas não essenciais, ou seja, devem ser atendidas sempre que possível.

Na Figura 10 é apresentado um fragmento de uma solução para a o PPT, a qual representa uma escala para os motoristas e cobradores da empresa. Esta é constituída por uma lista de jornadas. A Jornada 1 do tipo Pegada simples começa às 15:12 horas e termina às 00:20 horas tendo, portanto, uma duração de 9:08 horas. Como a segunda tarefa apresenta a linha final 2104, que pertence ao grupo 2, e a terceira tarefa inicia na linha 5201 que também é do mesmo grupo, houve dessa forma uma troca de linhas permitida. Pelo fato das três tarefas serem realizadas em veículos diferentes ocorrem, portanto, 2 trocas de veículos. A Jornada 2 do tipo Dupla Pegada começa às 06:04 horas e termina às 16:30 horas, com uma duração de 7:24 horas. O ponto final da primeira tarefa é diferente do ponto inicial da segunda tarefa, logo a Jornada 2 possui uma troca de pontos permitida. A Jornada 2 é do tipo Dupla pegada porque o intervalo entre a primeira e a segunda tarefa é de 3:02 horas e, portanto, superior a 2 horas. A Jornada 3 é inválida e do tipo Pegada simples. Para esta jornada, o horário de início da segunda tarefa é inferior ao horário de fim da primeira, dessa maneira essa jornada apresenta 0:51 horas de sobreposição, entre outras inviabilidades.

Jornada 1 -> (Tipo: Pegada simples)							
Número do Veículo	Horário Inicial	Ponto Inicial	Linha Inicial/ Grupo	Horário Final	Ponto Final	Linha Final/ Grupo	Folga Acum.
30	15:12	6	2104/2	17:52	6	2104/2	5
20	19:00	6	2104/2	22:00	11	2104/2	5
56	22:10	11	5201/2	00:20	0	5201/2	5

Tempo Trabalhado: 9:08 Tempo ocioso: 1:33 Hora extra: 1:58
Troca de veículos: 2 Troca de linhas permitida: 1

Jornada 2 -> (Tipo: Dupla pegada)							
Número do Veículo	Horário Inicial	Ponto Inicial	Linha Inicial/ Grupo	Horário Final	Ponto Final	Linha Final/ Grupo	Folga Acum.
101	06:04	0	9206/3	08:33	2	9206/3	5
90	11:35	11	9206/3	14:00	11	9206/3	5
90	14:05	11	9206/3	16:30	11	9206/3	5

Tempo Trabalhado: 7:24 Tempo ocioso: 0:20 Hora extra: 0:44
Troca de veículos: 1 Troca de pontos permitida: 1

Jornada 3 -> (Tipo: Pegada simples (Inválida))							
Número do Veículo	Horário Inicial	Ponto Inicial	Linha Inicial/ Grupo	Horário Final	Ponto Final	Linha Final/ Grupo	Folga Acum.
69	14:04	13	8207/3	16:39	13	8207/3	5
19	15:48	6	2104/2	18:28	6	2104/2	5
19	18:45	6	2104/2	21:50	0	2104/2	5

Tempo Trabalhado: 7:46 Tempo ocioso: 0:32 Hora extra: 0:36
Troca de veículos: 1 Sobreposição: 0:51 Troca de pontos proibida: 1
Troca de linhas proibida: 1

Figura 10: Solução para o Problema de Programação de Tripulações

4 Método de Solução do PPT

Apresenta-se neste capítulo a metodologia utilizada para resolver o PPT. Na seção 4.1 é descrita a forma de representação da solução do problema. As estruturas de vizinhança desenvolvidas para explorar o espaço de soluções são apresentadas na seção 4.2. A seção 4.3 apresenta a função de avaliação. Na seção 4.4 mostra-se o procedimento de geração da solução inicial. Os algoritmos propostos são apresentados na seção 4.5. Finalmente, na seção 4.6, apresenta-se um método de geração completa de colunas para o PPT de forma a validar as metodologias heurísticas desenvolvidas.

4.1 Representação

Os elementos básicos para a formação das jornadas são as tarefas (Figura 11). Essas agregam as informações: número da tarefa, que é um identificador único da mesma, o grupo a qual as viagens da tarefa pertencem, horário inicial, horário final, linha inicial, linha final, ponto inicial, ponto final, folga acumulada (isto é, o período ocioso entre as viagens agrupadas na tarefa) e o número do veículo.

TAREFA	
Número	Grupo
Horário Inicial	Horário Final
Linha Inicial	Linha Final
Ponto Inicial	Ponto Final
Folga Acumulada	Número do Veículo

Figura 11: Modelagem de uma tarefa

As jornadas, conforme ilustrado na Figura 12, são constituídas por uma lista de tarefas. Para cada tarefa contabiliza-se:

(a) Sobreposição, que representa a quantidade de tempo, em minutos, em que duas ou mais tarefas são realizadas simultaneamente;

(b) Horas Extras, que representa a quantidade de tempo, em minutos, que excedem a duração normal de uma jornada;

(c) Horas Excedentes, que representa a quantidade de tempo, em minutos, que excedem a duração máxima de uma jornada;

(d) Ociosidade, que representa a quantidade de tempo, em minutos, relativamente à duração de uma jornada, em que não há realização de viagens;

(e) Tempo entre Jornadas, que representa a quantidade de tempo, em minutos, que falta para a diferença entre o horário final da jornada em um dia e o horário inicial da mesma no dia subseqüente alcançar 11 (onze) horas;

(f) Troca de Veículos, que representa o número de vezes em que há troca de veículos na jornada;

(g) Troca de Pontos Proibida, que representa o número de vezes em que ocorreu troca de pontos proibida;

As informações Troca de Pontos Permitida, Troca de Linhas Proibida e Troca de Linhas Permitida são calculadas como no item (g) e de acordo com as restrições homônimas definidas no capítulo 3.

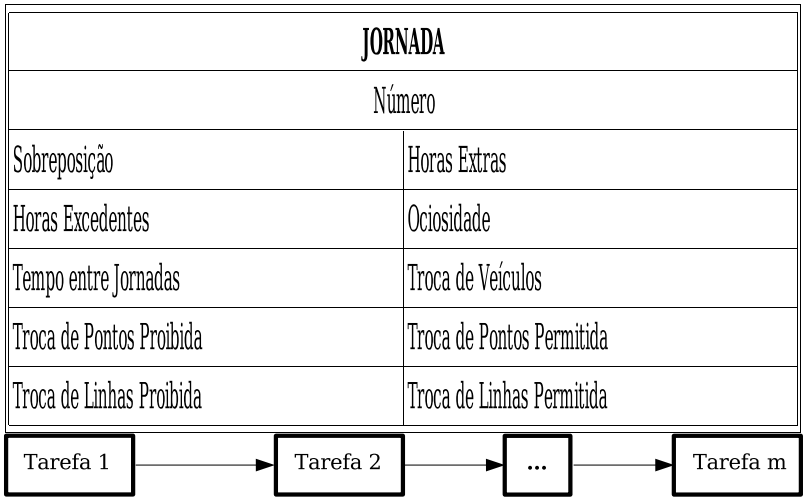

Figura 12: Modelagem de uma jornada

Uma solução para o problema é formada por uma lista de jornadas, conforme ilustra a Figura 13.

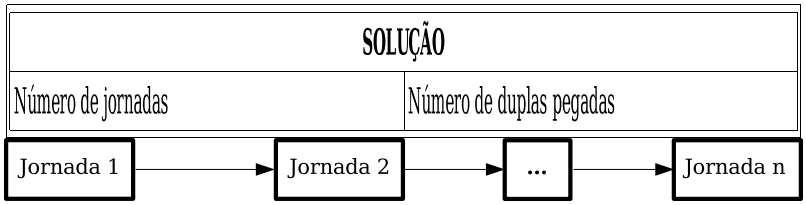

Figura 13: Representação da solução do PPT

4.2 Estruturas de Vizinhança

Foram desenvolvidos dois tipos de movimentos para definir a vizinhança $N(s)$ de uma dada solução s visando à exploração do espaço de soluções (Figura 14). O primeiro caracteriza-se pela realocação de uma tarefa de uma determinada jornada para outra jornada. O conjunto de todas as soluções s' geradas a partir de s através de movimentos de realocação define a vizinhança $N^{(R)}(s)$.

O segundo movimento é caracterizado pela troca de tarefas entre duas jornadas. O conjunto de todas as soluções s' geradas a partir de s através de movimentos de troca define a vizinhança $N^{(T)}(s)$.

Uma solução s' é dita vizinha de s se for obtida desta a partir de um movimento de realocação de tarefas ou de troca de tarefas, isto é, $s' \in N(s) = N^{(R)}(s) \bigcup N^{(T)}(s)$.

A Figura 14 ilustra a geração de vizinhos utilizando os dois tipos de movimento desenvolvidos. Em (a) mostra-se um vizinho da solução s gerado pelo movimento de realocação da tarefa "C" da jornada i para a jornada j. Já em (b), mostra-se um vizinho de s gerado pela troca das tarefas "D" e "G" entre as jornadas i e j.

4.3 Função de Avaliação

A função de avaliação considerada neste trabalho, a qual deve ser minimizada, é baseada na penalização dos requisitos não atendidos, conferindo aos requisitos essenciais um peso maior que aos requisitos não-essenciais. Uma solução s é avaliada com base na seguinte função f:

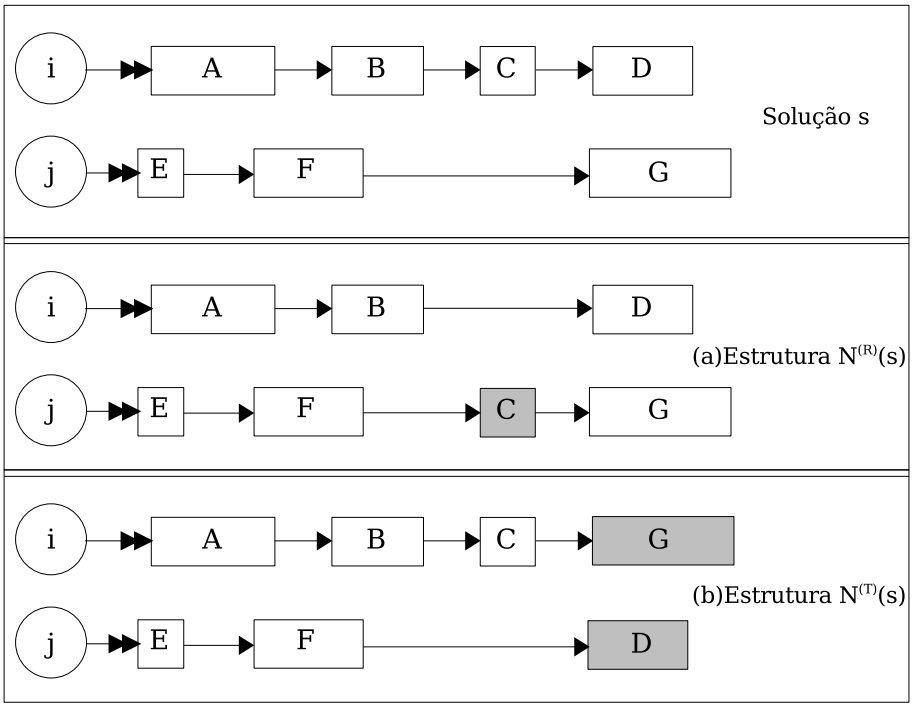

Figura 14: Estruturas de Vizinhança

$$f(s) = \sum_{i=1}^{n} \Big(\alpha_1 \times quantTempoOcioso_i + \alpha_2 \times quantHoraExtra_i + \alpha_3 \times numTrPPerm_i + \\ + \alpha_4 \times numTrLPerm_i + \alpha_5 \times numTrVeiculos_i + \beta_1 \times quantExcTempoTrab_i + \\ + \beta_2 \times quantSobrep_i + \beta_3 \times numTrPProib_i + \beta_4 \times numTrLProib_i + \\ + \beta_5 \times quantTempoEJornInsuf_i \Big) + \beta_6 \times TotalDuplaPeg \tag{4.1}$$

onde:

(i) n é o número de jornadas;

(ii) α_i e β_i são as penalidades aplicadas aos requisitos não essenciais e essenciais respectivamente;

(iii) $quantTempoOcioso_i$ é a quantidade total de ociosidade, em minutos, na jornada i e α_1 a penalização desse quesito;

(iv) $quantHoraExtra_i$ é a quantidade de horas extras, em minutos, na jornada i e α_2 a penalização para esse quesito;

(v) $numTrPPerm_i$ é o número de vezes em que ocorreu troca de pontos permitida durante a jornada i e α_3 a penalização desse quesito;

(vi) $numTrLPerm_i$ é o número de vezes em que ocorreu troca de linhas permitida durante a jornada i e α_4 a penalização correspondente;

(vii) $numTrVeiculos_i$ é o número de vezes em que ocorreu troca de veículos durante a jornada i e α_5 a penalização para esse quesito;

(viii) $quantExcTempoTrab_i$ é a quantidade de horas excedentes, em minutos, na jornada i e β_1 a penalização correspondente;

(ix) $quantSobrep_i$ é a quantidade de tempo, em minutos, em que duas ou mais tarefas são realizadas simultaneamente na jornada i e β_2 a penalização para esse quesito;

(x) $numTrPProib_i$ é o número de vezes em que ocorreu troca de pontos proibida durante a jornada i e β_3 a penalização desse quesito;

(xi) $numTrLProib_i$ é o número de vezes em que ocorreu troca de linhas proibida durante a jornada i e β_4 a penalização desse quesito;

(xii) $quantTempoEJornInsuf_i$ é a quantidade de tempo, em minutos, que falta para que a diferença entre o horário final da jornada i em um dia e o horário inicial no dia subseqüente atinja o tempo mínimo entre jornadas de 11 (onze) horas. β_5 é a penalização correspondente;

(xiii) $TotalDuplaPeg$ é a quantidade de jornadas do tipo Dupla Pegada que supera um certo número, previamente definido pela empresa. β_6 é a penalização correspondente;

4.4 Procedimento de geração da solução inicial (IC)

A solução inicial é obtida a partir do particionamento do bloco de veículos nas oportunidades de troca, o que irá gerar um conjunto de tarefas. Esse procedimento é semelhante àquele descrito no capítulo 3, exceto que o tempo para troca de tripulações é de no mínimo 5 minutos.

O procedimento, denominado IC (*Intelligent Construction*), começa pela primeira viagem do bloco do veículo e procede pelas demais até encontrar um intervalo entre viagens de no mínimo 5 minutos. Então, as viagens analisadas nesse passo são agrupadas em tarefas. Em cada passo cada bloco de veículo é analisado independentemente dos demais. O procedimento é repetido até que sejam analisados todos os blocos de veículos. Ao final do procedimento tem-se um conjunto de tarefas. A partir desse conjunto de tarefas serão formadas as jornadas, conforme explicado a seguir.

Para minimizar o número de inviabilidades na solução inicial, as tarefas a serem agrupadas em jornadas pertencem ao mesmo veículo. Para essa finalidade é calculada a duração do bloco do veículo e verificado se o mesmo é do tipo pegada simples ou dupla pegada. Se o bloco do veículo for do tipo pegada simples e a duração do mesmo for igual ou inferior a 9:10 horas (duração máxima da jornada de trabalho do tipo pegada simples) é formada uma única jornada. Caso a duração seja superior a 9:10 horas e inferior a 18:20 horas são formadas duas jornadas. Se a duração for superior a 18:20 formam-se três jornadas. Dessa forma o requisito *horas excedentes* não é violado. No caso de o bloco do veículo ser do tipo dupla pegada a duração máxima da jornada é de 8:40 horas e o procedimento é semelhante ao descrito para o caso de pegada simples.

4.5 Algoritmos propostos

Na fase preliminar deste trabalho foram desenvolvidas duas versões da Busca Tabu visando a comparação com o método VNS-RTL proposto por Souza et al. (2004) e descrito na seção 2.4, página 21. A primeira versão, denominada BT-FI-IC e a segunda, denominada BT-BI-IC, são descritas nas seções 4.5.1 e 4.5.2 respectivamente.

Num segundo momento, feita a comparação entre as versões BT-FI-IC e BT-BI-IC, a versão mais robusta, conforme pode-se verificar na seção 5.2, página 52, foi então aprimorada com um mecanismo de Relaxação Adaptativa, resultando no método BTAR-FI-IC descrito na seção 4.5.3.

Por fim, foi desenvolvido o método BTPR-FI-IC, descrito na seção 4.5.4, o qual combina o método BT-FI-IC com um procedimento de reconexão por caminhos.

4.5.1 Busca Tabu com primeira melhora (BT-FI-IC)

Esta versão utiliza o procedimento IC de geração da solução inicial descrito na seção 4.4. Além disso, o melhor vizinho é procurado dentro de um percentual da vizinhança (50%), interrompendo-se a busca na vizinhança quando um movimento que melhore a solução corrente é encontrado. A Busca Tabu procede então para a próxima iteração. A idéia de reduzir o espaço de busca é devido ao fato de a iteração do método de Busca Tabu ser custosa computacionalmente neste problema. A interrupção da busca neste método, conhecida como primeira melhora (ou *First Improvement - FI*), é uma idéia aplicada comumente pelos pesquisadores visando à redução do tempo computacional de exploração da vizinhança corrente.

4.5.2 Busca Tabu com melhor vizinho (BT-BI-IC)

Esta versão utiliza o procedimento IC de geração da solução inicial descrito na seção 4.4. Além disso, procura-se o melhor vizinho começando com a exploração de um percentual reduzido da vizinhança (5%) e quando ocorre *BTMax* iterações sem melhora, esse percentual é aumentado por um fator de 10%, até o máximo de 50%. É importante destacar que todo o percentual dessa vizinhança é analisado (*Best Improvement - BI*), independentemente de já se ter uma solução de melhora. O objetivo nesta estratégia é baratear o custo de exploração da vizinhança corrente de uma solução e aumentar seu tamanho somente se não houver melhora durante o processo de busca.

4.5.3 Busca Tabu com primeira melhora e Relaxação Adaptativa (BTAR-FI-IC)

Esta versão corresponde ao algoritmo BT-FI-IC acrescida de um mecanismo de Relaxação Adaptativa (*Adaptive Relaxation - AR*). Utilizou-se um procedimento semelhante àquele descrito por Schaerf (1996), onde os pesos para cada fonte de inviabilidade são ajustados dinamicamente, como proposto originalmente por Gendreau, Hertz & Laporte (1994). Para cada fonte de inviabilidade i o peso β_i é multiplicado por um fator σ_i que varia de acordo com o seguinte esquema:

1. No início da busca $\sigma_i \leftarrow 1$.
2. A cada k movimentos:
   - se todas as k soluções visitadas são factíveis em relação à restrição i então $\sigma_i \leftarrow \sigma_i/\gamma$;
   - se todas as k soluções visitadas são infactíveis em relação à restrição i então $\sigma_i \leftarrow \sigma_i \times \gamma$;
   - se algumas soluções são factíveis e algumas outras são infactíveis, considerando a restrição i, então σ_i permanece inalterado.

O parâmetro γ é randomicamente selecionado, a cada vez, no intervalo $[1,8, 2,2]$, conforme proposto por Schaerf (1996).

Cada valor de σ_i é limitado por duas constantes $\sigma_{i,min}$ e $\sigma_{i,max}$, o que evita que a Relaxação Adaptativa incremente/decremente indefinidamente os pesos para restrições que são sempre insatisfeitas/satisfeitas.

Na Figura 15 é apresentado o pseudo-código do algoritmo BTAR-FI-IC. Nesta figura, *IteracaoAtualizacao*() indica quando os pesos dinâmicos devem ser atualizados.

Algoritmo `BTAR-FI-IC`

1: $s \leftarrow GerarSolucaoInicial()$;
2: $s^* \leftarrow s$;
3: $ListaTabu \leftarrow \emptyset$;
4: **repita**
5: Selecione a vizinhança corrente CN, tal que $CN \in \{N^{(R)}(s), N^{(T)}(s)\}$;
6: Defina um subconjunto $V \subseteq CN$;
7: $melhorMovimento \leftarrow movimentoRandomico(V)$;
8: $melhorCustoDinamico \leftarrow \infty$;
9: **para todo** Movimento $m \in V$ **faça**
10: **se** ($f(s \oplus m) < f(s^*)$) **então**
11: $melhorMovimento \leftarrow m$;
12: **interromper**;
13: **senão**
14: **se** ($m \notin ListaTabu$) **então**
15: **se** ($f(s \oplus m) < f(s)$) **então**
16: $melhorMovimento \leftarrow m$;
17: **interromper**;
18: **senão**
19: **se** $(CustoPorPesosDinamicos(s \oplus m) < melhorCustoDinamico)$ **então**
20: $melhorMovimento \leftarrow m$;
21: $melhorCustoDinamico \leftarrow CustoPorPesosDinamicos(s \oplus m)$;
22: **fim se**
23: **fim se**
24: **fim se**
25: **fim se**
26: **fim para**
27: $s \leftarrow s \oplus melhorMovimento$;
28: **se** $f(s) < f(s^*)$ **então**
29: $s^* \leftarrow s$;
30: **fim se**
31: $AtualizarListaTabu()$;
32: **if** $IteracaoAtualizacao()$ **then** $AtualizarPesosDinamicos()$;
33: **até que** Critério de parada seja alcançado;
34: **return** s^*;

Figura 15: Algoritmo de Busca Tabu com Relaxação Adaptativa

4.5.4 Busca Tabu com primeira melhora e Reconexão por Caminhos (BTPR-FI-IC)

Esta versão corresponde ao algoritmo BT-FI-IC acrescido com um mecanismo de Reconexão por Caminhos (*Path Relinking - PR*). Seguiu-se a proposta apresentada na seção 2.6.3, página 28, aplicando-se a reconexão por caminhos como uma estratégia de intensificação sempre que se atinge um determinado número de iterações sem melhora na melhor solução ($IterAtivRC$). O procedimento de reconexão por caminhos é aplicado $IteracoesRC$ vezes, incorporando-se jornadas da solução guia na solução inicial. Explora-se apenas uma trajetória entre a solução inicial e a solução guia.

O funcionamento deste mecanismo é melhor ilustrado na Figura 16. Nesta figura, cada linha da matriz, que representa uma solução, constitue uma jornada e cada elemento da matriz $(1, \ldots, 9)$ é uma tarefa.

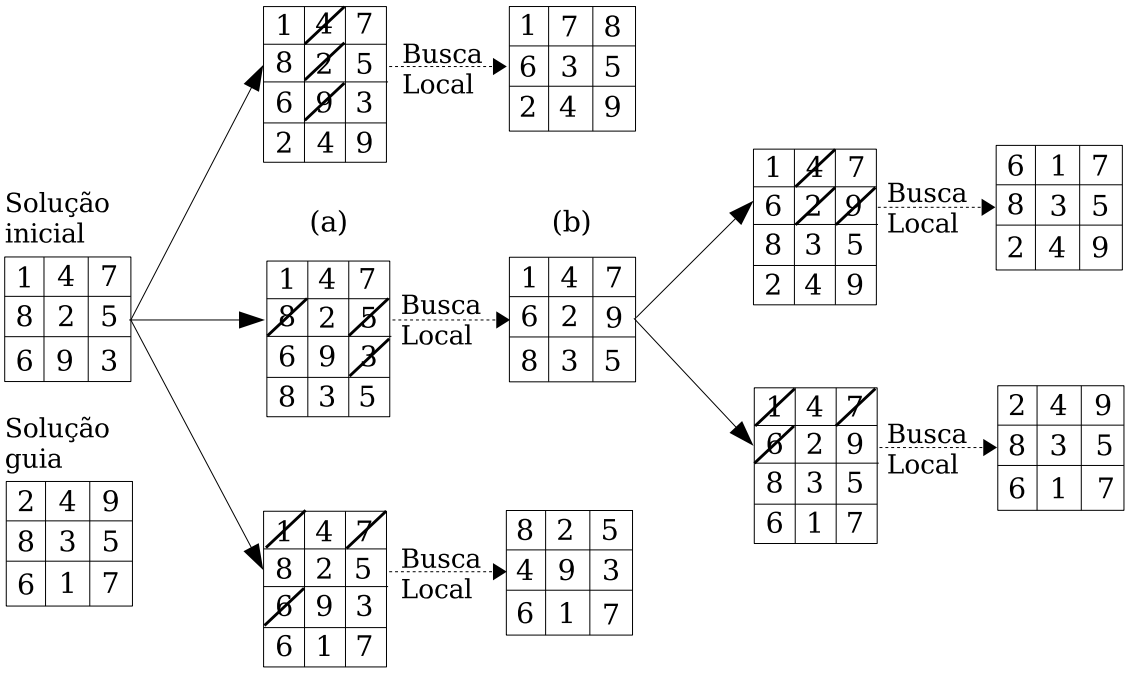

Figura 16: O mecanismo de intensificação por reconexão por caminhos

Para computar a diferença $\Delta(s_1, s_2)$ entre as soluções inicial e guia, é verificado se cada jornada da solução guia está na solução inicial. Para tornar este passo mais otimizado, utilizou-se o seguinte esquema de comparação entre jornadas:

1. Compara-se, inicialmente, a função de avaliação de cada jornada envolvida. Se

forem diferentes, conclui-se que as jornadas são diferentes. Caso contrário, passa-se para o passo seguinte;

2. Compara-se o número de tarefas de cada jornada envolvida. Se forem diferentes, conclui-se que as jornadas são diferentes. Caso contrário, analisa-se o passo seguinte;

3. Compara-se tarefa por tarefa de cada jornada envolvida. Se houver alguma tarefa diferente, conclui-se que as jornadas são diferentes; caso contrário, as jornadas são iguais.

Após ser computada a diferença entre as soluções inicial e guia, incorpora-se na solução inicial cada jornada presente na diferença. Com isso deve-se fazer a consistência das tarefas, eliminando-se as tarefas redundantes, confome constata-se em (a). Cada solução intermediária é então submetida a uma busca local para ser avaliada (b). O procedimento prossegue a partir da melhor solução intermediária encontrada.

Todas as versões de Busca Tabu desenvolvidas utilizam uma lista tabu de tamanho dinâmico. Para tal, o valor tabu *tenure* é selecionado aleatoriamente em um intervalo $\{minTabuTenure, \cdots, maxTabuTenure\}$. A cada iteração, apenas a vizinhança $N^{(R)}(s)$ ou apenas a vizinhança $N^{(T)}(s)$ é explorada. A primeira é selecionada com probabilidade pNR e a segunda com probabilidade $1 - pNR$.

4.6 Método de Geração Completa de Colunas

De forma a validar as metodologias baseadas em Busca Tabu apresentadas na seção 4.5, o PPT foi modelado através de um modelo de particionamento de conjuntos. Para a aplicação desse modelo são geradas todas as possíveis jornadas factíveis de trabalho, cada qual representando uma coluna. As colunas são geradas por meio de todas as combinações possíveis e factíveis entre tarefas. Nesse modelo, a cada tarefa está associada uma linha, a qual deve ser coberta por uma única coluna. O objetivo é encontrar, a partir do conjunto total de colunas, o subconjunto de menor custo que cubra todas as linhas.

Para a formulação matemática do PPT abordado sejam:

- Dados de entrada:

  $a_{ij} = \begin{cases} 1 & \text{se a tarefa } i = 1, \ldots, m \text{ é coberta pela jornada } j = 1, \ldots, \text{n} \\ 0 & \text{em caso contrário} \end{cases}$

  $a_{m+1,j} = \begin{cases} 1 & \text{se a jornada } j \text{ é do tipo dupla pegada} \\ 0 & \text{em caso contrário} \end{cases}$

  $NDP =$ Número máximo permitido de duplas pegadas

NJ = Número máximo permitido de jornadas

c_j = Custo da jornada j

- Variáveis de decisão:

  $x_j = 1$ se a jornada j for escolhida e 0 caso contrário.

A seguinte função deve ser minimizada:

$$\sum_{j=1}^{n} c_j x_j \tag{4.2}$$

Sujeito a:

$$\sum_{j=1}^{n} a_{ij} x_j = 1 \qquad \forall i = 1, \ldots, m \tag{4.3}$$

$$\sum_{j=1}^{n} a_{m+1,j} x_j \leq NDP \tag{4.4}$$

$$\sum_{j=1}^{n} x_j \leq NJ \tag{4.5}$$

$$x_j \in \{0, 1\} \qquad \forall j = 1, \ldots, n \tag{4.6}$$

As restrições (4.3) especificam que cada tarefa deve ser coberta por exatamente uma jornada. A restrição (4.4) indica que o número de duplas pegadas não deve ultrapassar um determinado número previamente definido. A restrição (4.5) especifica que o número de jornadas não pode superar o número máximo permitido de jornadas. As restrições (4.6) indicam que cada jornada pode ser escolhida uma única vez. O custo c_j da jornada j é calculado segundo a função de avaliação apresentada na seção 4.3, página 41.

5 Resultados e Discussão

Neste capítulo são apresentados os resultados dos testes computacionais realizados com os algoritmos desenvolvidos.

Para testar as metodologias desenvolvidas foram consideradas 11 instâncias-teste, cujas características principais estão descritas na seção 5.1. Para gerar os números pseudo-aleatórios necessários a essas metodologias utilizou-se o algoritmo *Mersenne Twister*, proposto por Matsumoto & Nishimura (1998), o qual produz números pseudo-aleatórios uniformes com período de $2^{19937} - 1$.

Inicialmente, são comparados os desempenhos das versões BT-FI-IC e BT-BI-IC do método de Busca Tabu (apresentadas nas seções 4.5.1 e 4.5.2, respectivamente), com o método VNS-RTL proposto por Souza et al. (2004) e apresentado na seção 2.4. Esses algoritmos foram aplicados à instância da empresa e os resultados são apresentados e discutidos na seção 5.2. Também nessa seção é feita uma análise de distribuição de probabilidade empírica de se alcançar um dado valor alvo (isto é, encontrar uma solução com um custo no mínimo tão bom quanto o valor alvo) em função do tempo.

A seguir, partiu-se para a solução das instâncias-teste pela aplicação do LINGO *Solver*, versão 7.0. O otimizador, que utiliza o procedimento *branch-and-bound* para explorar o espaço de soluções, é aplicado ao modelo de particionamentos de conjuntos descrito na seção 4.6, página 48. Observa-se que os dados de entrada do modelo são todas as possíveis jornadas factíveis e seus respectivos custos. Os resultados são apresentados e discutidos na seção 5.3.

Posteriormente, a versão BT-FI-IC do Método de Busca Tabu, a qual apresentou melhor desempenho quando comparada com os algoritmos BT-BI-IC e VNS-RTL, foi comparada com a versão BTAR-FI-IC apresentada na seção 4.5.3, a qual inclui um mecanismo de relaxação adaptativa. Os resultados são apresentados e discutidos na seção 5.4.

Para essas duas metodologias, analisou-se a distribuição de probabilidade empírica de se alcançar um dado valor alvo em função do tempo em diferentes instâncias. Os resultados são apresentados e discutidos na seção 5.5.

Finalmente, comparou-se a versão BTAR-FI-IC (de melhor desempenho que a versão

BT-FI-IC) com a versão BTPR-FI-IC, apresentada na seção 4.5.4, a qual inclui um mecanismo de reconexão por caminhos. Os resultados são apresentados e discutidos na seção 5.6. Também nesta seção, essas versões são comparadas entre si com relação à distribuição de probabilidade empírica.

Todos os algoritmos foram implementados em C++, usando a versão 3.2.2 do compilador gcc, com o flag -O3. Os experimentos foram realizados em um microcomputador Pentium IV, 2.6 GHz com 512 MBytes de RAM e sob o sistema operacional Linux.

A Tabela 2 resume as características dos algoritmos desenvolvidos.

Tabela 2: Características dos algoritmos

Algoritmo	**Características**
BT-FI-IC	Busca Tabu com primeira melhora numa vizinhança reduzida (Seção 4.5.1, página 44)
BT-BI-IC	Busca Tabu com melhor vizinho numa vizinhança inicialmente reduzida (Seção 4.5.2, página 45)
BTAR-FI-IC	Busca Tabu com primeira melhora numa vizinhança reduzida e Relaxação Adaptativa (Seção 4.5.3, página 45)
BTPR-FI-IC	Busca Tabu com primeira melhora numa vizinhança reduzida e Reconexão por Caminhos (Seção 4.5.4, página 47)

5.1 Instâncias-teste

Foram consideradas 12 instâncias-teste. A primeira delas, denominada PPT-Enterprise, é uma instância real referente a dados de uma empresa do Sistema de Transporte Público da cidade de Belo Horizonte (MG), do mês de março de 2002. As demais instâncias são derivadas dessa a partir do particionamento de suas linhas.

As características das mesmas encontram-se ilustradas na Tabela 3.

Para cada instância derivada a partir da instância real, foi gerado um conjunto de todas as possíveis colunas por enumeração completa, ou seja, foram consideradas todas as combinações de tarefas que resultassem em jornadas viáveis. A Tabela 4 apresenta o número de colunas geradas para cada instância.

Tabela 3: Características das instâncias

Instância	**Número de Tarefas**	**Número de Veículos**
PPT-Enterprise	733	101
PPT-L201-14	14	1
PPT-L321-19	19	2
PPT-L101-23	23	2
PPT-L1170-23	23	2
PPT-L2152-43	43	6
PPT-L4150-63	63	9
PPT-L5201-75	75	12
PPT-L8208-78	78	12
PPT-L9206-82	82	14
PPT-L2104-94	94	17
PPT-L8207-101	101	18

Tabela 4: Número de colunas geradas por instância

Instância	**Número de Colunas**
PPT-Enterprise	19.042.417
PPT-L201-14	12018
PPT-L321-19	8071
PPT-L101-23	30817
PPT-L1170-23	5671
PPT-L2152-43	9592
PPT-L4150-63	6301
PPT-L5201-75	5435
PPT-L8208-78	11253
PPT-L9206-82	10762
PPT-L2104-94	7437
PPT-L8207-101	12014

5.2 Comparação com o Método VNS-RTL

Inicialmente, foi feita uma bateria preliminar de testes para calibrar os diversos parâmetros das versões BT-BI-IC e BT-FI-IC. Em ambas, os parâmetros adotados foram $BTMax = 500$ (número máximo de iterações sem melhora no valor da melhor solução), $minTabuTenure = 55$, $maxTabuTenure = 60$ e $pNR = 0,8$ (probabilidade de escolha da vizinhança $N^{(R)}(s)$). Os valores de penalidade para os requisitos não essenciais encontram-se na Tabela 5 e para os requisitos essenciais, na Tabela 6. Esses valores são os mesmos utilizados em Souza et al. (2004).

A seguir, as versões BT-BI-IC e BT-FI-IC foram comparadas com o método VNS-RTL. Para tal finalidade foram realizados 20 testes para cada versão, cada qual partindo

Tabela 5: Penalizações dos requisitos não essenciais para comparação com o método VNS-RTL

Penalização	**Valor**
α_1 (Tempo Ocioso)	40
α_2 (Hora Extra)	60
α_3 (Troca de Pontos Permitida)	300
α_4 (Troca de Linhas Permitida)	300
α_5 (Troca de Veículos)	5000

Tabela 6: Penalizações dos requisitos essenciais para comparação com o método VNS-RTL

Penalização	**Valor**
β_1 (Horas Excedentes)	9000
β_2 (Sobreposição)	5000
β_3 (Troca de Pontos Proibida)	13000
β_4 (Troca de Linhas Proibida)	13000
β_5 (Tempo entre Jornadas)	5000
β_6 (Total de duplas pegadas)	9000

de uma semente diferente de números aleatórios, em um tempo de execução de uma hora. De forma a promover uma comparação justa com o método VNS-RTL, considerou-se uma nova versão, denominada BT-FI, que utiliza o mesmo procedimento de construção de uma solução inicial do método VNS-RTL, descrito na seção 2.4. Este último método foi executado em um microcomputador com a mesma configuração descrita anteriormente, porém sob o sistema operacional Windows XP.

A Tabela 7 apresenta o melhor valor encontrado para a função de avaliação e o desvio médio em relação ao valor da melhor solução encontrada em todos os testes, isto é:

$$desvio = \frac{ValorMedio - MelhorValor}{MelhorValor} \tag{5.1}$$

Tabela 7: Desempenho dos algoritmos BT-BI-IC e BT-FI-IC em relação ao método VNS-RTL

Método	**Melhor Valor**	**Desvio**
VNS-RTL	1.188.980	29,15%
BT-BI-IC	955.560	1,7%
BT-FI-IC	956.460	1,21%
BT-FI	955.640	1,53%

Pela Tabela 7, observa-se que a versão BT-FI-IC foi a que produziu o menor desvio, reduzindo o valor da melhor solução produzida pelo método VNS-RTL em 19,6%.

A Tabela 8 apresenta algumas características da melhor solução produzida por cada método, comparando-as com a solução obtida pela empresa.

Tabela 8: Características das melhores soluções geradas pelos algoritmos

	Empresa	**VNS-RTL**	**BT-BI-IC**	**BT-FI-IC**	**BT-FI**
# Jornadas	219	219	218	219	217
Horas extras (hh:mm)	116:00	87:13	85:25	83:31	87:56
Trocas de Veículos	0	28	27	26	27
Trocas de Linhas	0	13	14	11	17

Pela Tabela 8 observa-se que, em relação à empresa, todos os métodos heurísticos produziram soluções com redução no número de horas extras. Adicionalmente, os métodos BT-BI-IC e BT-FI conseguiram reduzir o número de jornadas e, conseqüentemente, de tripulações necessárias. Essa redução foi possível devido à flexibilidade da operação, aumentando o número de trocas de veículos e de linhas.

A Figura 17 ilustra a evolução típica da melhor solução produzida pelos métodos BT-FI e VNS-RTL, os quais partem de uma mesma solução inicial, nos instantes iniciais da execução dos mesmos. Por esta figura e pelos resultados apresentados anteriormente, verifica-se que o método BT-FI, além de produzir resultados finais de melhor qualidade que o método VNS-RTL, gera soluções de melhor qualidade mais rapidamente.

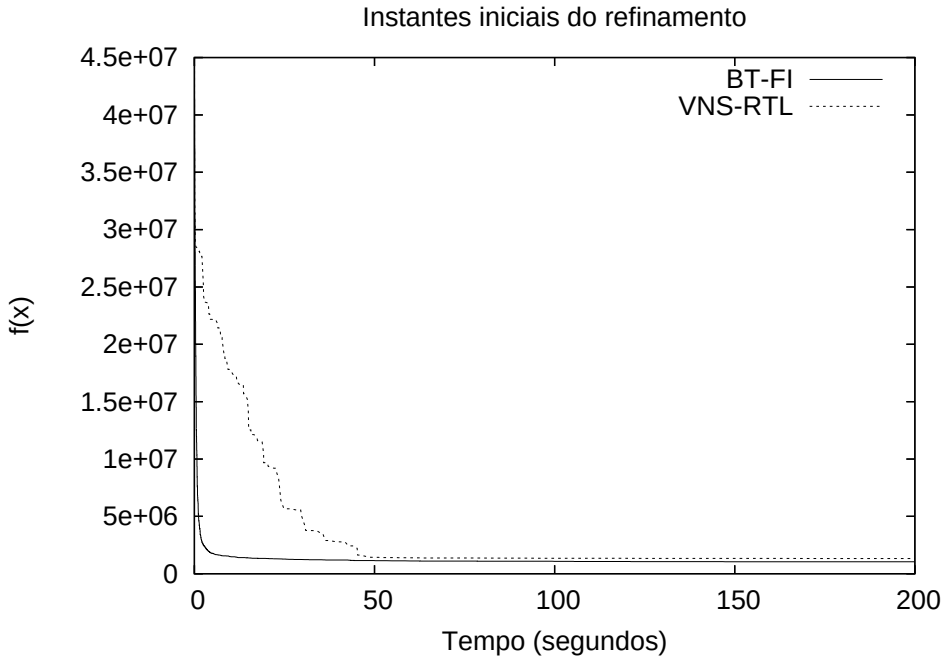

Figura 17: Evolução dos algoritmos BT-FI e VNS-RTL nos instantes iniciais do refinamento

No próximo conjunto de experimentos, verificou-se a distribuição de probabilidade empírica de alcançar um dado valor alvo (ou seja, encontrar uma solução com um custo

no mínimo tão bom quanto o valor alvo) em função do tempo para a instância da empresa. O valor alvo foi escolhido de forma que a versão mais lenta pudesse terminar em um tempo razoável. Os experimentos foram conduzidos conforme a proposta de Aiex, Resende & Ribeiro (2002). Os resultados foram plotados associando ao i-ésimo tempo de execução a probabilidade $p_i = (i - \frac{1}{2})/100$, gerando pontos $z_i = (t_i, p_i)$, para $i = 1, \ldots, 100$.

A Figura 18 apresenta os resultados para as versões BT-FI-IC e VNS-RTL, considerando um valor alvo igual a 2.000.000 relativo à instância PPT-Enterprise. Conforme mencionado no início desta seção, observa-se que os pesos considerados na função de avaliação são os mesmos adotados por Souza et al. (2004).

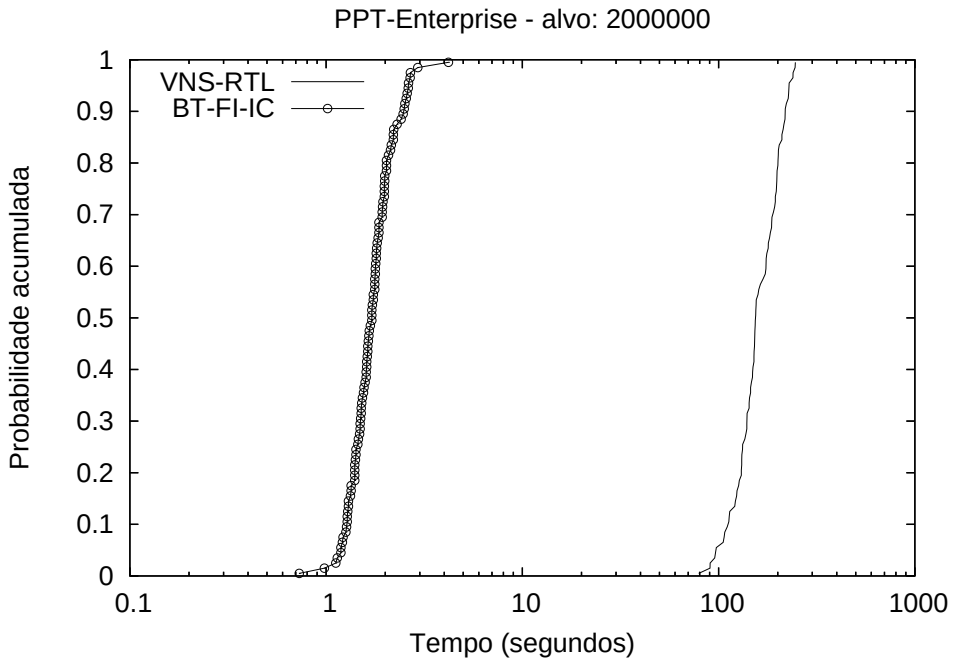

Figura 18: Distribuição de probabilidade empírica de encontrar um valor alvo em função do tempo para a instância PPT-Enterprise

Pode-se verificar pela Figura 18 que a versão BT-FI-IC alcança o valor alvo em um tempo significativamente menor que a versão VNS-RTL, qualquer que seja o valor de probabilidade.

5.3 Resolução das instâncias por geração completa de colunas

Com o objetivo de obter as soluções ótimas das instâncias-teste e validar as metodologias heurísticas desenvolvidas, foram geradas todas as possíveis jornadas factíveis de trabalho das instâncias-teste e seus respectivos custos, aplicando-se a esse conjunto o modelo de particionamento de conjuntos apresentado na seção 4.6, página 48.

Para resolver o PPT por esta metodologia, utilizou-se o modelador e otimizador LINGO, versão 7.0. A Tabela 9 mostra as soluções ótimas dessas instâncias e também o tempo gasto pelo otimizador, em segundos, para encontrá-las. Observa-se que não foi possível encontrar a solução ótima para a instância real da empresa, devido ao elevado número de colunas geradas.

Tabela 9: Solução ótima para cada instância e o tempo para encontrá-la

Instância	**Solução ótima**	**Tempo (seg)**
PPT-Enterprise	-	-
PPT-L201-14	32	122
PPT-L321-19	68	96
PPT-L101-23	32	442
PPT-L1170-23	59	82
PPT-L2152-43	103	153
PPT-L4150-63	226	162
PPT-L5201-75	452	187
PPT-L8208-78	263	1965
PPT-L9206-82	416	1953
PPT-L2104-94	639	426
PPT-L8207-101	707	2301

5.4 Avaliação das versões BT-FI-IC e BTAR-FI-IC

As versões BT-FI-IC e BTAR-FI-IC foram aplicadas ao conjunto de instâncias-teste para as quais se conhece as soluções ótimas e também à instância real obtida junto à empresa, para verificar a influência do mecanismo de Relaxação Adaptativa no método de Busca Tabu.

Para a versão BTAR-FI-IC utilizou-se o parâmetro $k = 10$, apresentado na seção 4.5.3, página 45, indicando de quantas em quantas iterações o procedimento de relaxação adaptativa é ativado.

Os valores adotados para as constantes $\sigma_{i,min}$ e $\sigma_{i,max}$ desse método, relativos à cada restrição, são apresentados na Tabela 10.

Tabela 10: Valores das constantes $\sigma_{i,min}$ e $\sigma_{i,max}$ para cada restrição

Requisito	$\sigma_{i,min}$	$\sigma_{i,max}$
Horas Excedentes (β_1)	0,01	7
Sobreposição (β_2)	0,01	10
Troca de Pontos Proibida (β_3)	0,01	3
Troca de Linhas Proibida (β_4)	0,01	3
Tempo entre Jornadas (β_5)	0,01	7
Total de duplas pegadas (β_6)	0,01	10

A Tabela 11 apresenta os valores utilizados para a penalização dos requisitos essenciais.

Tabela 11: Penalizações dos requisitos essenciais

Penalização	**Valor**
β_1 (Horas Excedentes)	40
β_2 (Sobreposição)	40
β_3 (Troca de Pontos Proibida)	50
β_4 (Troca de Linhas Proibida)	35
β_5 (Tempo entre Jornadas)	40
β_6 (Total de duplas pegadas)	45

A Tabela 12 apresenta os valores adotados para a penalização dos requisitos não-essenciais. Observa-se que a penalização dada ao requisito hora extra tem a ordem de grandeza dos pesos dos requisitos essenciais quando a quantidade de hora extra ultrapassa 120 minutos.

Tabela 12: Penalizações dos requisitos não essenciais

Penalização	**Valor**
α_1 (Tempo Ocioso ≤ 120)	$\lceil 0,347 \times quantTempoOcioso^{0,8} \rceil$
α_1 (Tempo Ocioso > 120)	18
α_2 (Hora Extra ≤ 120)	$\lceil 1,357 \times quantHoraExtra^{0.6} \rceil$
α_2 (Hora Extra > 120)	40
α_3 (Troca de Pontos Permitida)	5
α_4 (Troca de Linhas Permitida)	5
α_5 (Troca de Veículos)	19

Na Tabela 12, os valores dados às penalizações dos requisitos que medem a quantidade de ociosidade e de hora extra seguem uma função que procura diferenciar as jornadas de acordo com essas quantidades. Para quantidades menores as penalizações são relativamente mais baixas e diferentes entre si. Para quantidades maiores, as penalizações são um pouco mais elevadas e há uma diferença menos acentuada entre elas. O símbolo $\lceil . \rceil$ indica a função menor inteiro ou função teto.

A Figura 19 ilustra o comportamento da função de penalização relativa ao tempo ocioso.

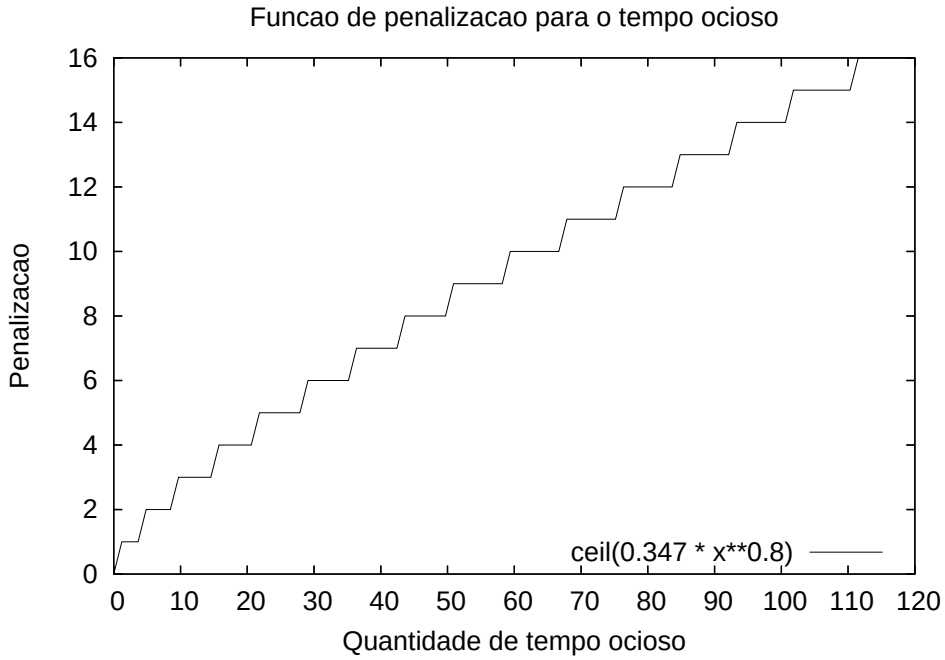

Figura 19: Comportamento da função de penalização para o tempo ocioso

Para cada instância foi realizado um conjunto de experimentos variando o tamanho da lista tabu. Realizaram-se 10 execuções, cada qual partindo de uma semente diferente de números aleatórios. O critério de parada adotado (linha 7 da Figura 5, página 25) foi o tempo de processamento. Para a instância PPT-Enterprise esse tempo foi fixado em uma hora, tal como em Souza et al. (2004). Para as demais instâncias, utilizou-se um tempo de execução sempre 30% inferior ao tempo gasto pelo LINGO *Solver* para encontrar a solução ótima. Os tempos de execução são apresentados na Tabela 13. Os resultados dos desempenhos dos métodos BT-FI-IC e BTAR-FI-IC são apresentados na Tabela 14.

Tabela 13: Tempo de execução para cada instância

Instância	**Tempo (seg)**
PPT-Enterprise	3600
PPT-L201-14	86
PPT-L321-19	68
PPT-L101-23	310
PPT-L1170-23	58
PPT-L2152-43	108
PPT-L4150-63	114
PPT-L5201-75	131
PPT-L8208-78	1376
PPT-L9206-82	1368
PPT-L2104-94	299
PPT-L8207-101	1611

Tabela 14: Distância média das melhores soluções conhecidas, para cada instância, nas versões BT-FI-IC e BTAR-FI-IC

BT-FI-IC					
	minTabuTenure-maxTabuTenure				
Instância	31-39	36-44	40-50	45-55	**Média**
PPT-Enterprise	4,74	4,71	4,53	4,58	4,64
PPT-L201-14	0,00	0,00	0,00	0,00	0,00
PPT-L321-19	0,00	0,00	0,00	0,00	0,00
PPT-L101-23	0,00	0,00	0,00	0,00	0,00
PPT-L1170-23	0,00	0,00	0,00	0,00	0,00
PPT-L2152-43	0,00	0,00	0,00	0,00	0,00
PPT-L4150-63	18,85	18,10	17,08	12,88	16,73
PPT-L5201-75	0,00	0,00	0,00	0,00	0,00
PPT-L8208-78	0,00	0,00	0,00	0,00	0,00
PPT-L9206-82	8,00	8,00	8,00	7,62	7,91
PPT-L2104-94	0,20	0,02	0,06	0,13	0,10
PPT-L8207-101	1,49	1,49	1,49	1,49	1,49
Média	2,77	2,69	2,60	2,22	2,57
BTAR-FI-IC					
	minTabuTenure-maxTabuTenure				
Instância	31-39	36-44	40-50	45-55	**Média**
PPT-Enterprise	1,93	1,77	1,98	2,00	1,92
PPT-L201-14	0,00	0,00	0,00	0,00	0,00
PPT-L321-19	0,00	0,00	0,00	0,00	0,00
PPT-L101-23	0,00	0,00	0,00	0,00	0,00
PPT-L1170-23	0,00	0,00	0,00	0,00	0,00
PPT-L2152-43	0,00	0,00	0,00	0,00	0,00
PPT-L4150-63	0,00	0,00	0,00	0,00	0,00
PPT-L5201-75	0,00	0,00	0,00	0,00	0,00
PPT-L8208-78	0,00	0,00	0,00	0,00	0,00
PPT-L9206-82	1,08	3,87	3,87	2,14	2,74
PPT-L2104-94	0,00	0,00	0,00	0,00	0,00
PPT-L8207-101	0,13	0,06	0,00	0,06	0,06
Média	0,26	0,47	0,49	0,35	0,39

Na Tabela 14 são apresentadas as distâncias médias, em porcentagem, do custo das soluções produzidas em relação às melhores soluções conhecidas. Pode-se constatar que para a versão BT-FI-IC, melhores resultados foram obtidos com intervalos tabu *tenure* mais elevados. A versão BTAR-FI-IC, por sua vez, obtém melhores resultados com quaisquer intervalos tabu *tenure*. Utilizando o intervalo 31-39, BTAR-FI-IC produz soluções, em média, distantes apenas 0,26% das melhores soluções conhecidas. O método BTPR-FI-IC obteve, para a instância da empresa, a melhor solução conhecida de custo igual a 3171.

Em vista desta análise, considera-se doravante que nos experimentos com as duas versões do método de Busca Tabu apenas os parâmetros que produzem os melhores resultados médios, ou seja, um intervalo tabu *tenure* igual a 45-55 para a versão BT-FI-IC e 31-39 para a versão BTAR-FI-IC são aplicados.

Uma visão diferente dos resultados dos experimentos previamente descritos é apresentada na Tabela 15. Os custos médios das soluções produzidas pelas versões propostas são comparados dentro do mesmo limite para o tempo de execução.

Tabela 15: Resultados médios em execuções com limites de tempo fixo para as versões BT-FI-IC e BTAR-FI-IC

Instância	**BT-FI-IC**	**BTAR-FI-IC**
PPT-Enterprise	3316,10	3232,10
PPT-L201-14	32,00	32,00
PPT-L321-19	68,00	68,00
PPT-L101-23	32,00	32,00
PPT-L1170-23	59,00	59,00
PPT-L2152-43	103,00	103,00
PPT-L4150-63	255,10	226,00
PPT-L5201-75	452,00	452,00
PPT-L8208-78	263,00	263,00
PPT-L9206-82	447,70	420,50
PPT-L2104-94	639,80	639,00
PPT-L8207-101	717,50	707,90

Conforme se pode observar pelas tabelas 15 e 9, a versão BTAR-FI-IC, descrita na seção 4.5.3, página 45, alcança como resultado médio o valor da solução ótima em 100% das execuções sobre as instâncias PPT-L4150-63 e PPT-L2104-94.

5.5 Distribuição de probabilidade empírica para as versões BT-FI-IC e BTAR-FI-IC

Neste conjunto de experimento verificou-se a distribuição de probabilidade empírica de alcançar um dado valor alvo (ou seja, encontrar uma solução com um custo no mínimo tão bom quanto o valor alvo) em função do tempo em diferentes instâncias. Os valores alvo foram escolhidos de forma que a versão mais lenta pudesse terminar em um tempo razoável. Os experimentos foram conduzidos também conforme a proposta de Aiex, Resende & Ribeiro (2002). Os resultados foram plotados associando ao i-ésimo tempo de execução a probabilidade $p_i = (i-\frac{1}{2})/100$, gerando pontos $z_i = (t_i, p_i)$, para $i = 1, \ldots, 100$.

A Figura 20 apresenta os resultados para as versões BT-FI-IC e BTAR-FI-IC, considerando um valor alvo igual a 3450 relativo à instância PPT-Enterprise.

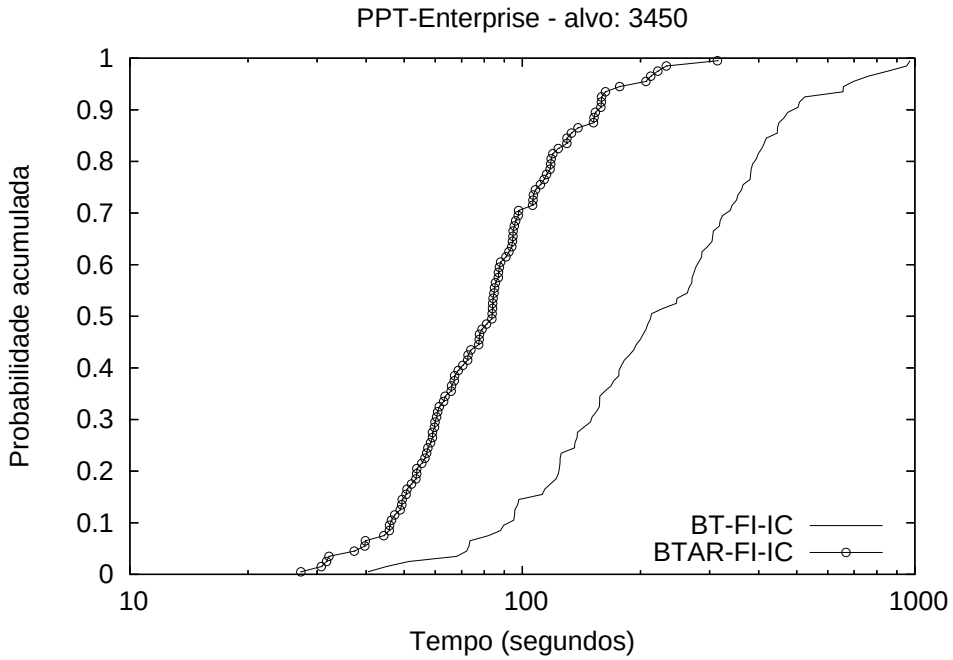

Figura 20: Distribuição de probabilidade empírica de encontrar um valor alvo em função do tempo para a instância PPT-Enterprise

Pode-se verificar, pela Figura 20, que para qualquer valor de probabilidade, a versão BTAR-FI-IC alcança o valor alvo em um tempo significativamente menor que a versão BT-FI-IC.

A Figura 21 ilustra a distribuição de probabilidade relativa à aplicação das versões BT-FI-IC e BTAR-FI-IC na instância PPT-L2104-94, considerando um valor alvo igual a seu ótimo, ou seja, 639.

Pela Figura 21 constata-se que a versão BT-FI-IC alcança o valor alvo apenas em pouco mais de 50% das execuções, ao passo que a versão BTAR-FI-IC sempre alcança o

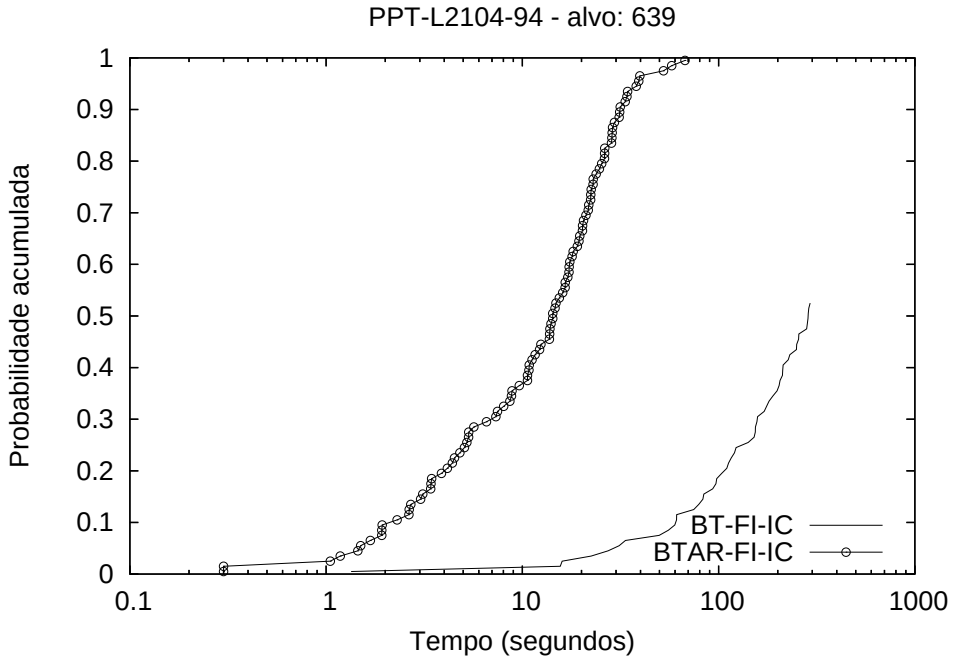

Figura 21: Distribuição de probabilidade empírica de encontrar um valor alvo em função do tempo para a instância PPT-L2104-94

valor alvo e em um tempo significativamente menor. Além disso, BTAR-FI-IC com 100% de probabilidade atinge o valor alvo em menos de 100 segundos enquanto o otimizador LINGO leva cerca de 426 segundos para atingir a otimalidade desta instância.

Na Figura 22 são apresentados os resultados para as versões BT-FI-IC e BTAR-FI-IC considerando um valor alvo igual ao ótimo da instância PPT-L4150-63, ou seja, 226.

Pela Figura 22 constata-se que a versão BT-FI-IC alcança o valor alvo apenas em menos de 40% das execuções, ao passo que a versão BTAR-FI-IC sempre alcança o valor alvo e em um tempo significativamente menor. Além disso, com 100% de probabilidade a BTAR-FI-IC atinge o valor alvo em menos de 10 segundos, enquanto o LINGO *Solver* leva cerca de 162 segundos para alcançar a solução ótima desta instância.

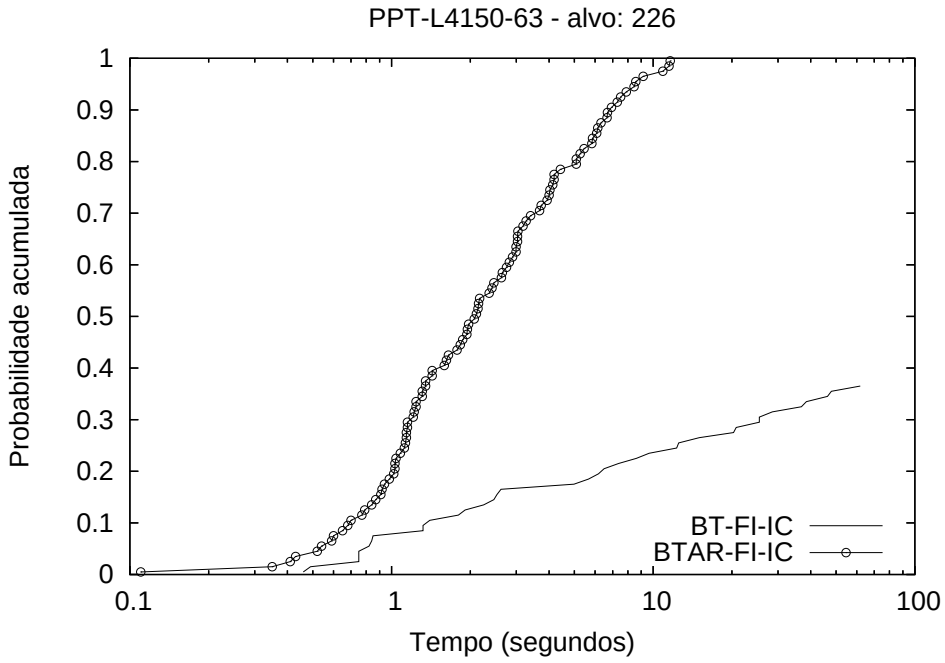

Figura 22: Distribuição de probabilidade empírica de encontrar um valor alvo em função do tempo para a instância PPT-L4150-63

5.6 Comparação entre as versões BTAR-FI-IC e BTPR-FI-IC

Nesta seção é feita a comparação de desempenho entre as versões BTAR-FI-IC, que se mostrou superior à versão BT-FI-IC, com a versão BTPR-FI-IC, a qual faz uso do mecanismo de reconexão de caminhos.

A versão BTPR-FI-IC foi aplicada ao conjunto de instâncias-teste para as quais são conhecidas as soluções ótimas e também à instância real obtida junto à empresa. As penalizações utilizadas foram as mesmas apresentadas na seção 5.4. Os parâmetros utilizados para a versão BTPR-FI-IC foram $IteracoesRC = 1$, $IterAtivRC = 700$, $TamConjElite = 5$ e $PercDif = 5$.

Para cada instância foi realizado um conjunto de experimentos variando o tamanho da lista tabu. Foram feitas 10 execuções, cada qual partindo de uma semente diferente de números aleatórios. O critério de parada adotado (linha 7 da Figura 5, página 25) foi o tempo de processamento. O tempo de execução para a instância PPT-Enterprise foi de uma hora, tal como nas comparações anteriores. Para as demais instâncias, utilizou-se um tempo de execução sempre 30% inferior ao tempo gasto pelo otimizador LINGO para encontrar a solução ótima. Os tempos de execução são apresentados na Tabela 16. Os resultados desta comparação são apresentados na Tabela 17.

Tabela 16: Tempo de execução para cada instância

Instância	**Tempo (seg)**
PPT-Enterprise	3600
PPT-L201-14	86
PPT-L321-19	68
PPT-L101-23	310
PPT-L1170-23	58
PPT-L2152-43	108
PPT-L4150-63	114
PPT-L5201-75	131
PPT-L8208-78	1376
PPT-L9206-82	1368
PPT-L2104-94	299
PPT-L8207-101	1611

Tabela 17: Distância média das melhores soluções conhecidas, para cada instância, nas versões BTAR-FI-IC e BTPR-FI-IC

BTAR-FI-IC					
	minTabuTenure-maxTabuTenure				
Instância	31-39	36-44	40-50	45-55	**Média**
PPT-Enterprise	1,93	1,77	1,98	2,00	1,92
PPT-L201-14	0,00	0,00	0,00	0,00	0,00
PPT-L321-19	0,00	0,00	0,00	0,00	0,00
PPT-L101-23	0,00	0,00	0,00	0,00	0,00
PPT-L1170-23	0,00	0,00	0,00	0,00	0,00
PPT-L2152-43	0,00	0,00	0,00	0,00	0,00
PPT-L4150-63	0,00	0,00	0,00	0,00	0,00
PPT-L5201-75	0,00	0,00	0,00	0,00	0,00
PPT-L8208-78	0,00	0,00	0,00	0,00	0,00
PPT-L9206-82	1,08	3,87	3,87	2,14	2,74
PPT-L2104-94	0,00	0,00	0,00	0,00	0,00
PPT-L8207-101	0,13	0,06	0,00	0,06	0,06
Média	0,26	0,47	0,49	0,35	0,39
BTPR-FI-IC					
	minTabuTenure-maxTabuTenure				
Instância	31-39	36-44	40-50	45-55	**Média**
PPT-Enterprise	2,00	2,02	1,62	1,76	1,85
PPT-L201-14	0,00	0,00	0,00	0,00	0,00
PPT-L321-19	0,00	0,00	0,00	0,00	0,00
PPT-L101-23	0,00	0,00	0,00	0,00	0,00
PPT-L1170-23	0,00	0,00	0,00	0,00	0,00
PPT-L2152-43	0,00	0,00	0,00	0,00	0,00
PPT-L4150-63	15,58	15,75	13,27	7,17	12,94
PPT-L5201-75	0,00	0,00	0,00	0,00	0,00
PPT-L8208-78	0,00	0,00	0,00	0,00	0,00
PPT-L9206-82	1,08	1,08	1,08	1,08	1,08
PPT-L2104-94	0,61	0,55	0,61	0,67	0,61
PPT-L8207-101	3,18	3,18	3,18	3,18	3,18
Média	1,87	1,88	1,65	1,16	1,64

Na Tabela 17 são apresentadas as distâncias médias, em porcentagem, do custo das soluções produzidas em relação às melhores soluções conhecidas. Pode-se constatar que para a versão BTPR-FI-IC, melhores resultados foram obtidos com intervalos tabu *tenure* mais elevados. A versão BTAR-FI-IC, por sua vez, obtém melhores resultados, com quaisquer intervalos tabu *tenure*. Utilizando o intervalo 31-39, BTAR-FI-IC produz soluções, em média, distantes apenas 0,26% das melhores soluções conhecidas. Entretanto, verifica-se que a reconexão por caminhos não trouxe vantagens significativas para a Busca Tabu implementada. Apenas para a instância PPT-L9206-82 verifica-se um ganho médio um pouco maior em relação à Busca Tabu com Relaxação Adaptativa. Observa-se,

também, que o método BTPR-FI-IC obteve a melhor solução conhecida para a instância da empresa, de custo igual a 3171.

Uma visão diferente dos resultados dos experimentos previamente descritos é apresentada na Tabela 18. Os custos médios das soluções produzidas pelas versões propostas são comparados dentro do mesmo limite de tempo de execução. Esta tabela considera os parâmetros que produzem os melhores resultados médios de cada versão, ou seja, um intervalo tabu *tenure* igual a 45-55 para a versão BTPR-FI-IC e 31-39 para a versão BTAR-FI-IC.

Tabela 18: Resultados médios em execuções com limites de tempo fixo para as versões BTAR-FI-IC e BTPR-FI-IC

Instância	**BTAR-FI-IC**	**BTPR-FI-IC**
PPT-Enterprise	3232,10	3226,80
PPT-L201-14	32,00	32,00
PPT-L321-19	68,00	68,00
PPT-L101-23	32,00	32,00
PPT-L1170-23	59,00	59,00
PPT-L2152-43	103,00	103,00
PPT-L4150-63	226,00	242,20
PPT-L5201-75	452,00	452,00
PPT-L8208-78	263,00	263,00
PPT-L9206-82	420,50	420,50
PPT-L2104-94	639,00	643,30
PPT-L8207-101	707,90	729,50

Conforme se pode extrair das tabelas 18 e 9, a versão BTAR-FI-IC alcança como resultado médio o valor da solução ótima em 100% das execuções feitas sobre as instâncias PPT-L4150-63 e PPT-L2104-94.

No próximo conjunto de experimentos, verificou-se a distribuição de probabilidade empírica de alcançar um dado valor alvo em função do tempo para a instância da empresa. O valor alvo foi escolhido de forma que a versão mais lenta pudesse terminar em um tempo razoável. Os experimentos também foram conduzidos conforme a proposta de Aiex, Resende & Ribeiro (2002). Os resultados foram plotados associando ao i-ésimo tempo de execução a probabilidade $p_i = (i - \frac{1}{2})/100$, gerando pontos $z_i = (t_i, p_i)$, para $i = 1, \ldots, 100$.

A Figura 23 apresenta um comportamento típico das duas versões com relação à probabilidade de se encontrar uma solução cujo custo seja igual ou inferior a um valor alvo. Nesta figura, relativa à instância PPT-Enterprise, o valor alvo considerado é 3450.

Nesta figura, mostra-se que até cerca de 90 segundos, a versão BTPR-FI-IC apresenta uma probabilidade maior de encontrar o valor alvo do que a versão BTAR-FI-IC. Acima

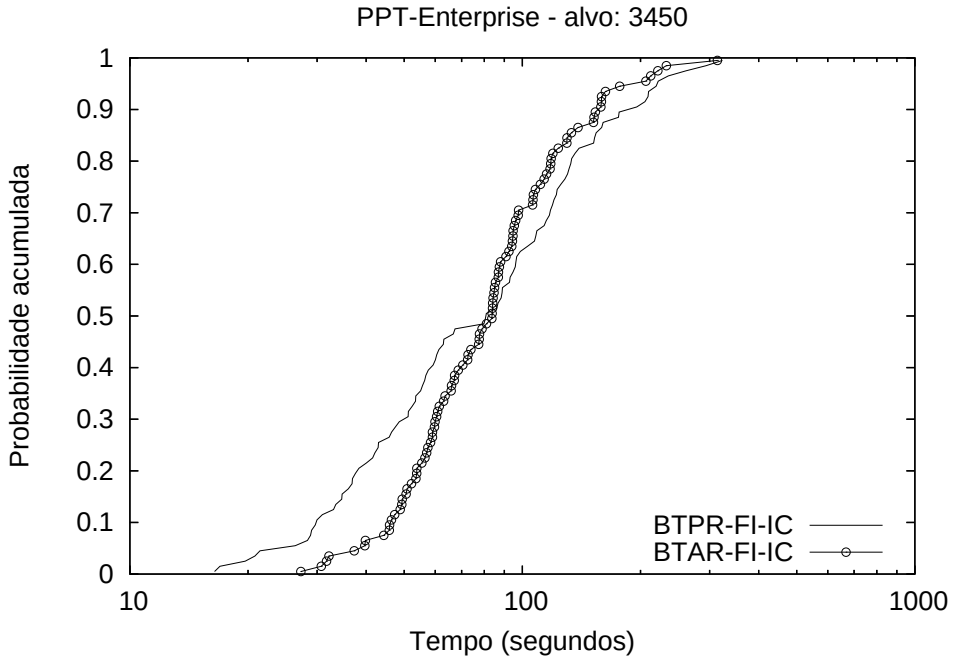

Figura 23: Distribuição de probabilidade empírica de encontrar um valor alvo em função do tempo para a instância PPT-Enterprise

de 90 segundos, a situação se inverte. Este comportamento pode se dever ao fato de que o mecanismo de reconexão por caminhos, por trabalhar com soluções elite e ativar precocemente a intensificação, consegue melhor desempenho no início da busca. Entretanto, com o decorrer da busca, o conjunto elite tende a ser mais uniforme com relação à qualidade das soluções ou mesmo ser atualizado com menos freqüência e, assim, as melhoras produzidas não são tão significativas. A Relaxação Adaptativa por sua vez, apesar de no início da busca não ter um desempenho tão bom quanto o da Reconexão por Caminhos, por equilibrar os mecanismos de intensificação e diversificação, tem capacidade de produzir melhoras mais significativas no final da busca.

6 Conclusões e trabalhos futuros

Este trabalho abordou o problema de programação de tripulações de ônibus urbano (PPT) de uma empresa que opera no sistema de tranporte público do município de Belo Horizonte (MG). Esse é um problema de grande importância no planejamento dos sistemas de tranporte, pois pela sua resolução as empresas do setor podem reduzir custos operacionais e aprimorar processos produtivos. Além deste estudo de caso, fez-se uma revisão de alguns dos principais trabalhos encontrados na literatura.

Para solucionar o PPT foram implementadas e avaliadas algumas versões da metaheurística Busca Tabu. A escolha dessa metaheurística é justificada pelo seu sucesso na abordagem de outros problemas combinatoriais, pelo baixo número de parâmetros a ser calibrados durante os experimentos e também pela sua boa robustez, quando comparado a outras metaheurísticas. A primeira versão desenvolvida, chamada BT-FI-IC, mostrou-se bastante robusta e rápida quando comparada ao método VNS-RTL proposto por Souza et al. (2004), que até então apresentava os melhores resultados para a instância fornecida pela empresa. Todos os métodos heurísticos produziram soluções com redução no número de horas extras, a qual representa uma das componentes que mais pesam na planilha de custos (BOUZADA, 2002).

Com o intuito de comparar a versão BT-FI-IC com um método exato foi utilizado um método de geração completa de colunas. Para tal comparação foram criadas 11 instâncias a partir de uma instância real da empresa. Para cada uma das instâncias, gerou-se um conjunto de todas as colunas possíveis por enumeração completa, ou seja, foram consideradas todas as combinações de tarefas que resultassem em jornadas viáveis. A cada um dos conjuntos de colunas aplicou-se o modelo de particionamento de conjuntos, sendo este resolvido pelo modelador e otimizador LINGO, versão 7.0. Para a instância fornecida pela empresa não foi possível encontrar a solução ótima devido ao elevado número de colunas geradas. A versão BT-FI-IC foi então executada sobre todas as instâncias-teste. Constatou-se pelos resultados que esta versão apresentava um desvio médio relativamente alto em relação ao valor da melhor solução conhecida para algumas das instâncias criadas.

A versão BT-FI-IC foi então aprimorada com um mecanismo de Relaxação Adaptativa. Tal versão, denominada BTAR-FI-IC, foi aplicada às instâncias-teste e comparou-se os resultados àqueles da versão BT-FI-IC. Os resultados mostraram uma melhora signi-

ficativa na robustez e rapidez com a inserção desse mecanismo. A versão BTAR-FI-IC produziu soluções, em média, distantes apenas 0,26% das melhores soluções conhecidas.

Também foi implementado um mecanismo de intensificação por Reconexão por Caminhos, tendo por base a versão BT-FI-IC. Esta versão, chamada BTPR-FI-IC, foi então aplicada às instâncias-teste. Esta foi a metodologia que obteve a melhor solução conhecida para a instância da empresa. Contudo, foi verificado que tal versão não apresentou resultados melhores quando comparada com a versão BTAR-FI-IC.

O mecanismo de Relaxação Adaptativa mostrou-se bastante eficiente na resolução do PPT, além de ser um algoritmo de mais simples implementação quando comparado ao complexo mecanismo de Reconexão por Caminhos. Desta forma, a inclusão de mecanismos de intensificação e diversificação em procedimentos baseados em Busca Tabu, pouco estudados para o PPT, é uma alternativa que pode trazer ganhos consideráveis.

Este trabalho mostrou-se relevante por resolver um problema real e auxiliar na difusão das técnicas de resolução do PPT ao cenário brasileiro, onde algumas empresas do setor de transporte ainda geram as escalas de trabalho de suas tripulações de forma manual. Ainda, demonstrou a grande eficácia dos métodos heurísticos, especialmente as metaheurísticas, em tratar problemas reais como o PPT, repletos de restrições operacionais, em tempos computacionais aceitáveis.

Como extensão deste trabalho pode-se propor, primeiramente, um procedimento híbrido, combinando a metaheurística Busca Tabu com uma metodologia baseada em programação matemática, para se tentar obter soluções de melhor qualidade. Poderia-se acionar na Busca Tabu, em um dado número de iterações sem melhora, um procedimento de programação matemática que resolvesse um subproblema de particionamento de conjuntos. Esse subproblema apareceria da desalocação, por um determinado critério, de um conjunto de jornadas e suas respectivas tarefas. A partir dessas tarefas seriam construídas todas as possíveis jornadas factíveis por enumeração completa. Esse subconjunto de jornadas seria submetido ao otimizador. Se houvesse melhora na solução global do problema, em função da resolução exata desse subproblema, então o procedimento deveria continuar com a desalocação de outro subconjunto de jornadas e respectivas tarefas. Este procedimento seria interrompido após um certo número de desalocações em que não mais houvesse melhora no valor da melhor solução gerada até então, situação na qual um procedimento de diversificação poderia ser acionado. Uma possível maneira de escolher as jornadas a serem desalocadas seria ordená-las pela sua duração, desalocando um certo número de jornadas de maior duração e outro certo número de jornadas de menor duração. O grupo de jornadas de maior duração seria escolhido, por exemplo, de forma aleatória dentre a primeira metade de jornadas de maior duração. De forma semelhante seria escolhido o grupo de jornadas de menor duração.

Pode-se propor ainda a implementação de outras metodologias heurísticas como me-

taheurísticas híbridas, heurísticas com procedimentos de mineração de dados, buscas locais com *ejection chains* e procedimentos de reconexão por caminhos mais sofisticados. Uma técnica de geração de colunas que tratasse as colunas implicitamente também poderia ser utilizada. Nesta metodologia, conhecido todo o conjunto A de possíveis colunas factíveis, partiria-se de um subconjunto seleto de colunas A^0, obtidas, por exemplo, por um procedimento heurístico. Esse subconjunto A^0 definiria um Problema Mestre Restrito (PMR) o qual seria otimizado. De posse dos valores das variáveis duais associadas às restrições do problema envolvendo todo o conjunto A de colunas, dar-se-ia início à resolução do problema escravo. Este subproblema consistiria em encontrar um conjunto N contendo uma ou mais colunas de A com custo reduzido negativo. Se $N \neq \emptyset$, far-se-ia $A^{i+1} = A^i \bigcup N$ e o processo se repetiria. Caso contrário, a solução ótima do problema mestre restrito corrente A^i seria a solução ótima do problema original.

Referências

AIEX, R. M.; RESENDE, M. G. C.; RIBEIRO, C. C. Probability distribution of solution time in GRASP: an experimental investigation. *Journal of Heuristics*, Boston, v. 8, p. 343–373, 2002.

ANBIL, R.; GELMAN, E.; PATTY, B.; TANGA, R. Recent advances in crew-pairing optimization at American Airlines. *Interfaces*, Linthicum, v. 21, n. 1, p. 62–74, 1991.

ANDERSSON, E.; HOUSOS, E.; KOHL, N.; WEDELIN, D. Crew pairing optimization. In: YU, G. (Ed.). *OR in Airline Industry*. Boston: Kluwer Academic Publishers, 1997. p. 1–31.

(ANTP) - ASSOCIAÇÃO NACIONAL DOS TRANSPORTES PÚBLICOS. *O Transporte na Cidade do Século XXI*. São Paulo, SP, 1999. Disponível em: <http://www.antp.org.br>.

BATTITI, R.; TECCHIOLLI, G. Parallel biased search for combinatorial optimization: genetic algorithms and Tabu. *Microprocessors and Microsystems*, Amsterdam, v. 16, n. 7, p. 351–367, 1992.

BEASLEY, J. E.; CAO, B. A tree search algorithm for the crew scheduling problem. *European Journal of Operational Research*, Amsterdam, v. 94, n. 3, p. 517–526, 1996.

BLAIS, J. .-Y.; LAMONT, J.; ROUSSEAU, J. .-M. The HASTUS vehicle and manpower scheduling system at the Société de transport de la Communauté urbaine de Montréal. *Interfaces*, Linthicum, v. 20, n. 1, p. 26–42, 1990.

BOUZADA, C. F. *Análises das despesas administrativas no custo do transporte coletivo por ônibus no município de Belo Horizonte*. Dissertação (Mestrado) — Escola do Governo, Fundação João Pinheiro, Belo Horizonte, 2002.

CABRAL, L. A. F.; PONTES, R. C.; SOUZA, M. J. F.; MACULAN, N. An heuristic approach for large scale crew scheduling problems at Rio-Sul Airlines. In: *Proceedings of the 40th AGIFORS, 40th International Symposium of the Agifors*. Istambul: [s.n.], 2000. p. 16.

CAO, B. *Algorithms for crew scheduling problems*. Tese (Doutorado) — Imperial College, Londres, 1992.

CAVIQUE, L.; REGO, C.; THEMIDO, I. Subgraph ejection chains and tabu search for the crew scheduling problem. *Journal of the Operational Research Society*, v. 50, p. 608–616, 1999.

CHEDDAD, H. *Algorithms for crew scheduling problems*. Tese (Doutorado) — Imperial College, Londres, 1987.

DADUNA, J. R.; BRANCO, I.; PAIXÃO, J. M. P. *Computer-Aided Transit Scheduling*. Berlim: Springer-Verlag, 1995. (Lecture Notes in Economics and Mathematical Systems, v. 430).

DADUNA, J. R.; VOSS, S. *Computer-Aided Scheduling of Public Transport*. Berlim: Springer Publishers, 2001. (Lecture Notes in Economics and Mathematical Systems, v. 505).

DADUNA, J. R.; WREN, A. *Computer-Aided Transit Scheduling*. Berlim: Springer-Verlag, 1988. (Lecture Notes in Economics and Mathematical Systems, v. 308).

DESAULNIERS, G.; DESROSIERS, J.; DUMAS, Y.; MARC, S.; RIOUX, B.; SOLOMON, M. M.; SOUMIS, F. Crew pairing at Air France. *European Journal of Operational Research*, Amsterdam, v. 97, n. 2, p. 245–259, 1997.

DESROCHERS, M.; ROUSSEAU, J. .-M. *Computer-Aided Transit Scheduling*. Berlim: Springer-Verlag, 1992. (Lecture Notes in Economics and Mathematical Systems, v. 386).

ELIAS, S. E. G. The use of digital computers in the economic scheduling for both man and machine in public transportation. Manhattan, Kansas, 1964.

EMDEN-WEINERT, T.; PROKSCH, M. Best practice simulated annealing for the airline crew scheduling problems. *Journal of Heuristics*, Boston, v. 5, n. 4, p. 419–436, 1999.

ERNST, A. T.; JIANG, H.; KRISHNAMOORTHY, M.; NOTT, H.; SIER, D. An integrated optimization model for train crew management. *Annals of Operations Research*, Basel, v. 108, n. 1, p. 211–224, 2001.

ERNST, A. T.; JIANG, H.; KRISHNAMOORTHY, M.; SIER, D. Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, Amsterdam, v. 153, n. 1, p. 3–27, 2004.

FORES, S.; PROLL, L.; WREN, A. An improved ILP system for driver scheduling. In: WILSON, N. H. M. (Ed.). *Computer-Aided Transit Scheduling*. Berlim: Springer-Verlag, 1999. p. 43–62.

FRELING, R. *Models and techniques for integrating vehicle and crew scheduling*. Tese (Doutorado) — Erasmus University, Rotterdam - Holanda, 1997.

GAREY, M. R.; JOHNSON, D. S. *Computers and Intractability - A Guide to the Theory of NP-Completeness*. New York: W. H. Freeman, 1979.

GENDREAU, M.; HERTZ, A.; LAPORTE, G. A tabu search heuristic for the vehicle routing problem. *Management science*, Linthicum, v. 40, n. 10, p. 1276–1290, 1994.

GERSHKOFF, I. Optimizing flight crew schedules. *Interfaces*, Linthicum, v. 19, n. 4, p. 29–43, 1989.

GLOVER, F. Future paths for integer programming and artificial intelligence. *Computers and Operations Research*, Amsterdam, v. 13, n. 5, p. 533–549, 1986.

GLOVER, F. Tabu search and adaptive memory programming - advances, applications and challenges. In: BARR, R.; HELGASON, R.; KENNINGTON, J. (Ed.). *Interfaces in Computer Science and Operations Research*. Boston Dordrecht London: Kluwer, 1996. p. 1–75.

GLOVER, F.; LAGUNA, M. *Tabu Search*. Boston Dordrecht London: Kluwer Academic Publishers, 1997.

HANSEN, P. The steepest ascent mildest descent heuristic for combinatorial programming. In: *Congress on Numerical Methods in Combinatorial Optimization*. Capri: [s.n.], 1986.

KWAN, A. S. K.; KWAN, R. S. K.; PARKER, M. E.; WREN, A. Producing train driver shifts by computer. In: ALLEN, J.; BREBBIA, C. A.; HILL, R. J.; SCIUTTO, G.; SONE, S. (Ed.). *Computers in Railways V, Vol. 1: Railways Systems Management*. Southamptom: Computational Mechanics Publications, 1996. p. 421–435.

KWAN, A. S. K.; KWAN, R. S. K.; PARKER, M. E.; WREN, A. Producing train driver schedules under different operating strategies. In: WILSON, N. H. M. (Ed.). *Computer-Aided Transit Scheduling*. Berlin: Springer-Verlag, 1999. p. 129–154.

KWAN, A. S. K.; KWAN, R. S. K.; WREN, A. Driver scheduling using genetic algorithms with embedded combinatorial traits. In: WILSON, N. H. M. (Ed.). *Computer-Aided Transit Scheduling*. Berlim: Springer-Verlag, 1999. p. 81–102.

KWAN, R. S. K.; KWAN, A. S. K.; WREN, A. Evolutionary driver scheduling with relief chains. *Evolutionary Computation*, v. 9, p. 445–460, 2001.

LAVOIE, S.; MINOUX, M.; ODIER, E. A new approach for crew pairing problems by column generation with an application to air transportation. *European Journal of Operational Research*, Amsterdam, v. 35, n. 1, p. 45–58, 1988.

LI, J.; KWAN, R. S. K. *A genetic algorithm with fuzzy comprehensive evaluation for driver scheduling*. Leeds, U. K., Junho 2000.

LOURENÇO, H. R.; PAIXÃO, J. P.; PORTUGAL, R. *Metaheuristics for the bus-driver scheduling problem*. Barcelona, Espanha, 2001. (Economics Working Papers, 304).

MATSUMOTO, M.; NISHIMURA, T. Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. *ACM Transactions on Modeling and Computer Simulation*, New York, v. 8, n. 1, p. 3–30, 1998.

MAURI, G. R. *Novas heurísticas para o problema de escalonamento de tripulações*. Dissertação (Mestrado) — Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2005.

MLADENOVIC, N.; HANSEN, P. Variable neighborhood search. *Computers and Operations Research*, v. 24, n. 11, p. 1097–1100, 1997.

OZDEMIR, H. T.; MOHAN, C. K. Flight graph based genetic algorithm for crew scheduling in airlines. *Information Sciences*, Amsterdam, v. 133, n. 3-4, p. 165–173, 2001.

PARKER, M. E.; SMITH, B. M. Two approaches to computer crew scheduling. In: WREN, A. (Ed.). *Computer Scheduling of Public Transport*. Amsterdam: North-Holland, 1981. p. 193–222.

PIMENTEL, A. L. G. *Uma abordagem heurística para a solução de problemas de recobrimento de conjuntos de grande porte, com aplicação à alocação de tripulações para companhias aéreas*. Tese (Doutorado) — Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2005.

RIBEIRO, C. C.; UCHOA, E.; WERNECK, R. F. A hybrid GRASP with perturbations for the Steiner problem in graphs. *INFORMS Journal on Computing*, Linthicum, v. 14, p. 228–246, 2002.

ROSSETI, I. C. M. *Estratégias seqüenciais e paralelas de GRASP com reconexão por caminhos para o problema de síntese de redes a 2-caminhos*. Tese (Doutorado) — Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2003.

ROUSSEAU, J. .-M. *Computer Scheduling of Public Transport 2*. Amsterdam: North-Holland, 1985.

SCHAERF, A. *Tabu search techniques for large high-school timetabling problems*. Amsterdam, 1996.

SHEN, Y.; KWAN, R. S. K. *Tabu search for time windowed public transport driver scheduling*. Leeds, U. K., Maio 2000.

SHEN, Y.; KWAN, R. S. K. Tabu search for driver scheduling. In: DADUNA, J. R.; VOSS, S. (Ed.). *Computer-Aided Scheduling of Public Transport*. Berlim: Springer-Verlag, 2001. p. 121–135.

SOUSA, J. F. A computer based interactive approach to crew scheduling. *European Journal of Operational Research*, Amsterdam, v. 55, n. 3, p. 382–393, 1991.

SOUZA, M. J. F.; CARDOSO, L. X. T.; SILVA, G. P.; RODRIGUES, M. M. S.; MAPA, S. M. Metaheurísticas aplicadas ao Problema de Programação de Tripulações no sistema de transporte público. *Tendências em Matemática Aplicada e Computacional*, São José dos Campos, v. 5, p. 12, 2004.

WILSON, N. H. M. *Computer-Aided Transit Scheduling*. Berlim: Springer-Verlag, 1999. (Lecture Notes in Economics and Mathematical Systems, v. 471).

WREN, A. *Computer Scheduling of Public Transport*. Amsterdam: North-Holland, 1981.

WREN, A. Heuristics ancient and modern: transport scheduling through the ages. *Journal of Heuristics*, Boston, v. 4, p. 87–100, 1998.

WREN, A.; KWAN, R. S. K.; KWAN, A. S. K. Hybrid genetic algorithms for scheduling bus and train drivers. In: WILSON, N. H. M. (Ed.). *Proceedings of the 2000 Congress on Evolutionary Computation CEC00*. San Diego: IEEE, 2000. p. 285–292.

WREN, A.; WREN, D. O. A genetic algorithm for public transport driver scheduling. *Computers and Operations Research*, Amsterdam, v. 22, n. 1, p. 101–110, 1995.

YUNES, T. H. *Problemas de escalonamento no transporte coletivo: programação por restrições e outras técnicas*. Dissertação (Mestrado) — Instituto de Computação, Universidade de Campinas, Campinas, 2000.