UNIVERSIDADE ESTADUAL DE MARINGÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

RUBENS ZENKO SAKIYAMA

Avaliação da Meta-heurística VNS para um Problema de Planejamento Operacional do Transporte Público

Maringá
2014

RUBENS ZENKO SAKIYAMA

Avaliação da Meta-heurística VNS para um Problema de Planejamento Operacional do Transporte Público

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Departamento de Informática, Centro de Tecnologia da Universidade Estadual de Maringá, como requisito parcial para obtenção do título de Mestre em Ciência da Computação

Orientador: Prof. Dr. Ademir Aparecido Constantino

Maringá
2014

Dados Internacionais de Catalogação na Publicação (CIP)
(Biblioteca Central - UEM, Maringá, PR, Brasil)

S158a

Sakiyama, Rubens Zenko
Avaliação da Meta-heurística VNS para um problema de planejamento operacional do transporte público / Rubens Zenko Sakiyama. -- Maringá, 2014.
86 f. : figs., tabs.

Orientador: Prof. Dr. Ademir Aparecido Constantino.
Dissertação (mestrado) - Universidade Estadual de Maringá, Centro de Tecnologia, Departamento de Informática, Programa de Pós-Graduação em Ciência da Computação, 2013.

1. Transporte público - Escalonamento. 2. Algoritmos - Problema de Escalonamento de Motoristas (PEM). 3. Algoritmos - Problemas de grande porte. 4. Algoritmos heurísticos. 5. Meta-heurística VNS. I. Constantino, Ademir Aparecido, orient. II. Universidade Estadual de Maringá. Centro de Tecnologia. Departamento de Informática. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDD 21.ed. 005.741
GVS-000990

FOLHA DE APROVAÇÃO

RUBENS ZENKO SAKIYAMA

Avaliação da meta-heurística VNS para um problema de planejamento operacional do transporte público

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Departamento de Informática, Centro de Tecnologia da Universidade Estadual de Maringá, como requisito parcial para obtenção do título de Mestre em Ciência da Computação pela Banca Examinadora composta pelos membros:

BANCA EXAMINADORA

Prof. Dr. Ademir Aparecido Constantino
Universidade Estadual de Maringá – DIN/UEM

Prof. Dr. Wesley Romão
Universidade Estadual de Maringá – DIN/UEM

Prof. Dr. Arinei Carlos Lindbeck da Silva
Universidade Federal do Paraná – MAT/UFPR

Aprovada em: 14 de fevereiro de 2014.
Local da defesa: Sala 101, Bloco C56, *campus* da Universidade Estadual de Maringá.

AGRADECIMENTOS

Agradeço primeiramente a Deus por ter me guiado nesta jornada com fé e saúde.

A minha esposa Marisa e aos meus filhos Lea e Marcelo que com muito carinho me apoiaram e incentivaram nesta jornada.

Ao meu orientador Prof. Dr. Ademir Aparecido Constantino pelos conhecimentos transmitidos apoio e paciência para o desenvolvimento deste trabalho.

Ao Prof. Dr. Wesley Romão pelas sugestões e comentários dispensados.

A Inês, secretária do PCC, sempre apoiando no preenchimento de formulários e alertando no cumprimento dos prazos.

A todos que direta ou indiretamente contribuíram para a realização deste trabalho.

Avaliação da Meta-heurística VNS para um Problema de Planejamento Operacional do Transporte Público

RESUMO

O Problema de Escalonamento de Motoristas (PEM) consiste em gerar uma escala de trabalho de motoristas, para cobrir uma escala de veículos com o menor custo, satisfazendo restrições impostas pelas leis trabalhistas, acordos sindicais e normas da empresa. Esse processo é uma etapa fundamental para o planejamento operacional de empresas do setor de transporte coletivo visto que o custo com motoristas atinge uma parcela significativa no custo global da empresa. Considerado como NP-Difícil, vários trabalhos abordam a resolução do PEM através de algoritmos heurísticos devido às limitações dos algoritmos exatos para tratar com instâncias de grande porte. O presente trabalho propõe uma abordagem para a solução do PEM envolvendo dois procedimentos de busca local em uma estrutura de vizinhança, denominados PCR e *k-swap*, de forma determinística e em conjunto com a meta-heurística VNS. Para validação da proposta foram utilizadas instâncias reais com mais de 2300 viagens e instâncias aleatórias extraídas das instâncias reais. Os experimentos realizados comprovam a eficiência da meta-heurística VNS para instâncias de grande porte, onde os resultados obtidos são comparados com resultados apresentados por outros trabalhos que utilizaram os procedimentos PCR e *k-swap* sem a utilização da meta-heurística VNS.

Palavras-chave: Transporte Público, Problema de Escalonamento de Motoristas, Problemas de Grande Porte, Heurística, Meta-heurística VNS.

Study of Heuristic Algorithms for a Operational Planning Problem of Public Transport

ABSTRACT

The Bus Driver Scheduling Problem (BDSP) consists to generate a set of drivers schedule to cover a set of vehicles schedule at the lowest cost, satisfying constraints imposed by labor laws, trade union agreements and company standards. This process is vital to the operational planning of public transportation companies since the drivers cost affects a significant portion of the overall cost of the company. Considered NP-Hard, several works address the resolution of PEM through heuristic algorithms due to the limitations of the exact algorithms to work with large instances. The present work propose a approach for solving the BDSP involving two local search procedures in a neighborhood structure, called PCR and k-swap, in a deterministic way and in conjunction with VNS meta-heuristic. To validate the work is proposed real instances with over 2300 travel and random instances extracted of real instances. The experiments demonstrated the efficacy of VNS meta-heuristic for large instances, where the present results are compared with results reported by other studies that used PCR and k-swap procedures without the use of VNS meta-heuristic.

Keywords: Public Transport, Bus Driver Scheduling Problem, Large Real-world Instances, Heuristic, VNS Metaheuristic.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AG: Algoritmo Genético

ATP: Algoritmo de Treinamento Populacional

BI: *Best Improvement*

BT: Busca Tabu

BVGP: Busca em Vizinhança de Grande Porte

BW: *Backward*

CLT: Consolidação das Leis do Trabalho

BDSP: *Bus Driver Scheduling Problem*

FI: *First Improvement*

FW: *Forward*

ILS: *Iterated Local Search*

OT: Oportunidade de troca

PA: Problema de Atribuição

PCR: Procedimento de Corte e Recombinações

PEM: Problema de escalonamento de motoristas

PET: Problema de escalonamento de tripulação

PEV: Problema de escalonamento de veículos

PL: Programação Linear

PLI: Programação Linear Inteira

PPT: Problema de programação de tripulação

SA: *Simulated Annealing*

VND: *Variable Neighborhood Descent*

VNS: *Variable Neighborhood Search*

GAMS: *General Algebraic Modelling System*

GRASP: *Greedy Randomized Adaptive Search Procedure*

SUMÁRIO

1 Introdução

1.1 Considerações Iniciais

O transporte coletivo de passageiros é um serviço essencial para a sociedade, pois além de gerar empregos diretos e indiretos, grande parte da população depende deste transporte para se locomover diariamente até seu trabalho.

O planejamento operacional no transporte coletivo envolve várias etapas ou atividades. A Figura 1.1 ilustra a sequência de atividades envolvidas neste planejamento de acordo com Prata (2010).

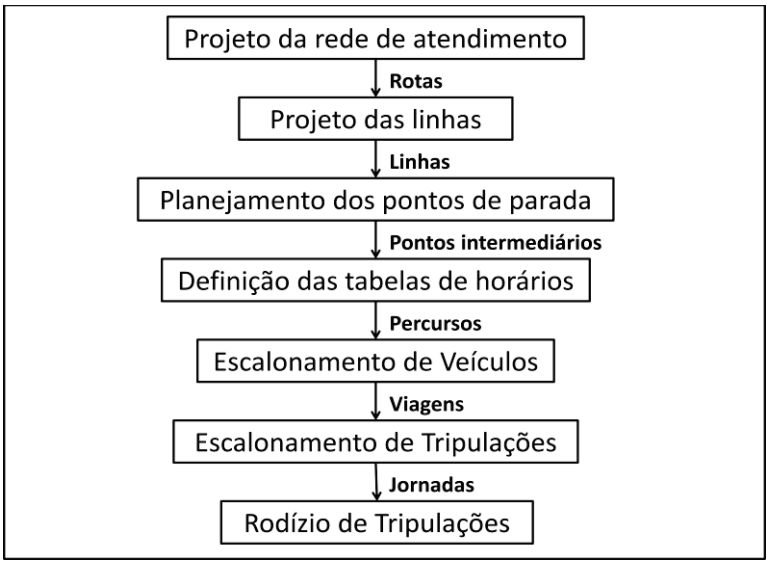

Figura 1.1: Planejamento do transporte público – Fonte: Adaptado de Prata (2010)

Em geral o órgão gestor de transporte do município realiza o projeto da rede de atendimento de acordo com a demanda. Em seguida realiza o projeto das linhas levando se em conta os pontos intermediários e seus horários. Estas informações são entregues às empresas de transporte coletivo que farão a alocação de seus veículos e motoristas para atender a estas linhas.

O problema do escalonamento de motoristas (PEM), também conhecido como problema de escalonamento de tripulação (PET) ou problema de programação de tripulação (PPT) é uma das etapas do planejamento do transporte público (Prata, 2010).

O PEM consiste em determinar a melhor programação diária dos condutores de forma a obedecer às restrições, minimizar custos operacionais e cumprir a escala de veículos para a tabela pré-determinada de horários.

1.2 Importância deste Trabalho

Os veículos e motoristas são os principais recursos de uma companhia de transporte público, e a forma como são alocados tem impacto direto na qualidade e no custo do serviço (Silva e Cunha, 2010). Além da redução direta de custos, há o benefício proporcionado aos funcionários, que poderiam cumprir jornadas menos exaustivas, aumentando a qualidade do serviço. Além disso, com jornadas de trabalho dentro da legislação evitam-se possíveis custos passivos (futuros) devido a ações trabalhistas (Calvi, 2005).

A importância de uma programação eficiente dos condutores está relacionada ao impacto desta nos custos totais de uma empresa de transporte coletivo, já que veículos e motoristas são os componentes que constituem a maior parte dos custos da empresa (Silva *et al*., 2002; Prata, 2010). De acordo com a URBS (Urbanização de Curitiba S/A), empresa de capital misto da qual a Prefeitura de Curitiba é a maior acionista, os **custos com pessoal e encargos representam 47,16% da composição tarifária**, conforme gráfico ilustrado pela Figura 1.2.

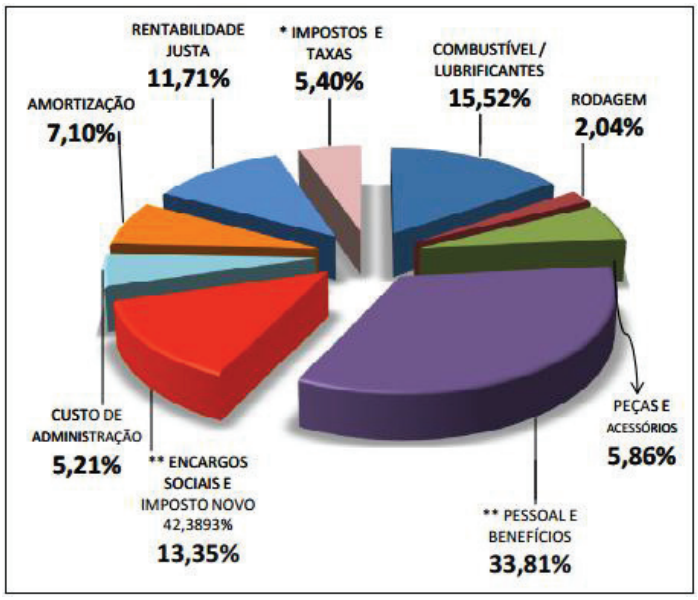

Figura 1.2: Planilha de Composição Tarifária – Fonte: URBS (2013)

O PEM tem sido extensivamente estudado na área de pesquisa operacional por mais de 40 anos e, desde 1975, uma série de eventos especializados na área (Daduna *et al.*, 1995; Daduna and Wren, 1988; Wilson, 1999; Voβ and Daduna, 2001) tem mostrado os progressos e novos desenvolvimentos. Estudos de métodos para solução do problema tem sido apresentados por Bodin *et al.* (1983), Wren e Rousseau (1993) e Wren (2004).

O PEM é considerado de difícil solução computacional, pertencendo a classe dos problemas NP-difícil, devido ao grande número de combinações possíveis para sua resolução e às diversas restrições que devem ser consideradas. As restrições impostas a este problema são provenientes da legislação trabalhista, de acordos com o sindicato da categoria e políticas da empresa de transporte.

Diante destas considerações, a maioria dos algoritmos apresentados para a resolução do PEM é baseada em heurísticas ou meta-heurísticas, ao invés da utilização de métodos exatos.

1.3 Objetivos

O objetivo deste trabalho é investigar um algoritmo heurístico para o problema de escalonamento de motoristas no transporte público.

Os objetivos específicos são:

a) Propor um algoritmo heurístico para a resolução do problema de escalonamento de motoristas utilizando os procedimentos de busca local PCR e *k-swap* de forma determinística e em conjunto com a meta-heurística VNS ;
b) Avaliar a influência dos operadores de busca local utilizando instâncias de dados reais e instâncias aleatórias obtidas dos dados reais, com ênfase as instâncias de grande porte;
c) Avaliar as soluções obtidas em relação a trabalhos anteriores que utilizaram os mesmos procedimentos de busca local PCR e *k-swap*.

1.4 Justificativas

A utilização de métodos heurísticos neste trabalho é justificada pelo fato do PEM ser um problema da classe NP-Difícil onde a utilização de métodos exatos está

limitada a instâncias de pequeno porte. Justifica-se a utilização da meta-heurística VNS apoiado na característica desta meta-heurística aceitar a utilização de procedimentos de busca local como PCR e *k-swap* em sua estrutura, procedimentos estes já utilizados com sucesso em outros trabalhos.

1.5 Limitações

O presente trabalho está limitado na resolução do PEM para as situações onde há somente dois pontos de troca: garagem e terminal.

1.6 Estrutura da Dissertação

Esta dissertação está organizada conforme a descrição dos capítulos a seguir. O Capítulo 2 apresenta a descrição do problema abordado detalhando conceitos básicos e nomenclaturas. Trabalhos relacionados a este serão apresentados também neste capítulo. O Capítulo 0 apresenta uma revisão da literatura descrevendo o Problema de Atribuição (PA), técnicas de busca local, a meta-heurística VNS e procedimentos de busca local em estruturas de vizinhança denominados PCR e *k-swap*. O Capítulo 0 apresenta a proposta deste trabalho com a descrição da função objetivo e dos procedimentos utilizados para a construção dos algoritmos propostos em duas abordagens: determinística e em conjunto com a meta-heurística VNS. O Capítulo 0 apresenta os resultados obtidos através dos experimentos realizados comparando a performance destas duas abordagens. Por fim, as conclusões são apresentadas no Capítulo 6.

2 Descrição do Problema

Este capítulo apresenta a definição, conceitos e nomenclaturas do Problema de Escalonamento de Motoristas (PEM), trabalhos desenvolvidos para a resolução deste problema utilizando métodos exatos e métodos heurísticos e trabalhos desenvolvidos para a resolução do Problema de Escalonamento de Veículos (PEV) e o Problema de Escalonamento de Motoristas (PEM) de forma integrada.

2.1 O Problema de Escalonamento de Motoristas

O problema de escalonamento de motoristas (PEM) consiste em gerar uma escala diária de trabalho, ou seja, um conjunto de jornadas que cubra toda uma escala de veículos predefinida com menor custo operacional atendendo às restrições do problema, que incluem a legislação imposta pela CLT, acordos coletivos com o sindicato da categoria e a política da empresa.

Segue abaixo alguns conceitos básicos e nomenclaturas relacionados ao PEM, que serão utilizados nesta dissertação:

- **Escala de veículos**: conjunto de viagens que cada um dos veículos deverá realizar durante um dia. É a entrada para resolução do PEM.
- **Bloco**: sequência de viagens a serem realizadas por um veículo, geralmente iniciando e terminando na garagem. O conjunto de blocos forma uma escala de veículos;
- **Motorista**: condutor do veículo, responsável pelo deslocamento dos passageiros;
- **Oportunidade de troca (OT)**: local e horário onde são permitidas trocas de motorista de um determinado veículo;
- **Viagem**: percurso entre duas oportunidades de trocas consecutivas. Durante uma viagem não poderá ocorrer troca de motorista. Em algumas literaturas é utilizado o termo tarefa;

- **Jornada**: conjunto de viagens realizadas por um único motorista durante seu dia de trabalho;
- **Escala de trabalho**: conjunto de jornadas que cobre toda a escala de veículos durante um dia. Também é denominada escala de motoristas. É a saída de resolução do PEM.

As principais características e a descrição das restrições envolvidas neste problema são apresentadas a seguir:

- A escala de veículos já está definida, através da solução gerada por um algoritmo que resolve o problema de escalonamento de veículos, e segue o planejamento realizado pela empresa;
- A programação das escalas de motoristas considera um horizonte de planejamento de um dia;
- São considerados dois possíveis pontos de troca entre os motoristas: a garagem da empresa e o terminal urbano;
- As trocas de veículos entre os motoristas são permitidas, mas devem ser feitas sem excessos;
- A jornada de trabalho normal do motorista tem duração de 7 (sete) horas e 20 (vinte) minutos. Caso a jornada realizada seja menor, ainda assim será pago conforme o tempo mínimo pago por uma jornada de trabalho, ou seja, 7 (sete) horas e 20 (vinte) minutos;
- O tempo máximo de horas extras trabalhadas não deve exceder 2 (duas) horas, ou seja, a jornada com horas extras poderá ter duração máxima de até 9 (nove) horas e 20 (vinte) minutos trabalhados;
- O valor das horas extras trabalhadas recebe um adicional de 50% sobre o valor pago pela hora normal;
- A duração máxima do trabalho contínuo é de 6 (seis) horas. Quando a duração do trabalho ultrapassar 6 (seis) horas, deverá ser concedido um intervalo para descanso;
- O intervalo para descanso deverá ter um tempo mínimo de 1 (uma) hora e 30 (trinta) minutos e máximo de 5 (cinco) horas;
- A extensão máxima da jornada, considerando o tempo trabalhado mais o tempo de folga, deve ser de até 13 (treze) horas. Isso ocorre, pois o intervalo

entre o fim da jornada de um dia e o início da jornada do outro dia deve ser de no mínimo 11 (onze) horas, assim considerando que um motorista poderá executar a mesma jornada no dia seguinte temos que a duração da jornada total não deve ser maior do que 13 (treze) horas;

- A hora trabalhada no período noturno, entre as 22 (vinte e duas) horas de um dia e às 5 (cinco) horas do dia seguinte, possui duração de 52,5 minutos;
- Sobre as horas trabalhadas no horário noturno haverá um adicional de 20% sobre o custo da jornada normal.

2.2 Trabalhos Relacionados

Devido ao grande número de combinações possíveis para a resolução do PEM e às diversas restrições que devem ser consideradas, o problema é de difícil solução computacional e pertence à classe NP-Difícil (De Leone *et al*., 2011). O uso de métodos exatos ainda é um fator crítico para instâncias de grande porte.

Apesar de encontrarmos trabalhos que utilizam técnicas de programação matemática, conforme ilustrado na Tabela 2.1, as instâncias utilizadas nestes trabalhos não ultrapassam dezenas ou algumas centenas de viagens. Silva *et al*. (2004) apresentam uma metodologia que formula o PEM como um modelo de particionamento utilizando o método *Simplex* para resolvê-lo. Para validar o método proposto foram realizados testes com instâncias reais (11 linhas contendo de 11 a 87 viagens), resolvendo um problema de particionamento para cada linha separadamente, com o uso do pacote LINGO.

Tabela 2.1: Métodos exatos para resolução do PEM

Autor	Ano	Técnica	No. viagens
Silva *et al.*	2004	Método Simplex + Pacote LINGO	11 a 87
Yunes *et al.*	2005	geração de colunas	125 e 246
De Leone *et al.*	2011	GAMS 2005 + Cplex 9.1.2	16 a 70

Yunes *et al*. (2005), utilizaram uma abordagem baseada no problema de cobertura de conjuntos e geração de colunas com o objetivo de minimizar o número de motoristas, utilizando duas instâncias reais, composta por duas linhas que atendem a região metropolitana de Belo Horizonte, sendo uma com 125 e outra com 246 viagens.

De Leone *et al*. (2011) propõem nova formulação matemática para o PEM considerando as restrições seguidas por empresas de transporte italianas. Para instâncias de pequeno e médio porte, contando com um número de viagens entre 16 e 70 viagens, a solução exata do modelo matemático proposto foi obtida utilizando as ferramentas GAMS (*General Algebraic Modelling System*) 2005 e Cplex 9.1.2, sendo que para a instância com 70 viagens a solução ficou restrita na minimização do número de jornadas não obtendo um custo viável do problema.

Na literatura encontramos diversas propostas de algoritmos heurísticos para o PEM. Marinho *et al*. (2004) apresentaram uma proposta baseada em Busca Tabu (BT) utilizando duas estruturas de vizinhança caracterizadas por dois tipos de movimentos. O primeiro movimento consiste em realocar uma tarefa de uma determinada jornada a outra jornada e o segundo em trocar tarefas entre duas jornadas. Silva *et al*. (2002) apresentam um algoritmo baseado na meta-heurística *Simulated Annealing* (SA). Souza *et al*. (2003) fazem uma continuidade de Silva *et al*. (2002), utilizando novas estruturas de vizinhança para o SA e para o Método VNS. Marinho *et al*. (2004) e Souza *et al*. (2003) utilizaram dados reais de empresas de transporte público da cidade de Belo Horizonte.

A Tabela 2.2 ilustra as propostas para resolução do PEM utilizando métodos heurísticos e/ou meta-heurísticas, onde ilustramos a dimensão das instâncias utilizadas pelo número de viagens realizadas.

Lourenço *et al.*(2001) propõem meta-heurísticas multiobjetivos usando Busca Tabu (BT) Multiobjetivo e Algoritmos Genéticos (AG) Multiobjetivo, combinados com GRASP (*Greedy Randomized Adaptive Search Procedure*) com a utilização de instâncias reais de seis empresas de transporte público de Portugal, participantes de um projeto de planejamento de transporte denominado GIST.

Dias *et al*. (2002) propõem um Algoritmo Genético que trata custos, número de jornadas, porcentagem da escala realizada por jornadas viáveis e duração média das jornadas como características importantes para obter uma solução de qualidade, utilizando 15 instâncias de duas maiores empresas de transporte urbano portuguesas sendo a menor com 102 viagens e a maior com 251 viagens.

Mauri e Lorena (2004) propõem uma metodologia interativa baseada na aplicação do Algoritmo de Treinamento Populacional (ATP) juntamente com Programação Linear (PL) com instâncias de 25 a 500 viagens geradas aleatoriamente a partir de instâncias reais de empresas de transporte público brasileiras.

Para a resolução de instâncias reais do PEM entre 19 e 138 viagens, obtidas de empresa de transporte público da cidade de Belo Horizonte, Santos (2007) apresentou um método que utiliza o Algoritmo Genético (AG) para a resolução dos subproblemas de geração de novas colunas, resolvidos por PL ou por PLI.

De Leone *et al.* (2011) propõem um algoritmo baseado na heurística GRASP para instâncias entre 80 e 400 viagens considerando as restrições seguidas por empresas de transporte italianas. Silva e Cunha (2010) também apresentaram um modelo de resolução do PEM baseado na heurística GRASP com busca local realizada pela técnica de Busca em Vizinhança de Grande Porte (BVGP) utilizando escalas de programação de veículos de uma empresa de transporte público de Belo Horizonte com um mínimo de 392 e um máximo de 945 viagens.

Gonçalves (2010) apresentou uma solução baseada na utilização das meta-heurísticas Busca Tabu e *Iterated Local Search* (ILS) com dados reais de uma empresa de transporte público da cidade de Belo Horizonte com número de viagens entre 27 (1 linha) e 515 (13 linhas).

Tabela 2.2: Propostas utilizando métodos heurísticos para resolução do PEM

Autor	Ano	Técnica	No. viagens
Lourenço *et al.*	2001	(Busca Tabu + Algoritmo Genético) + GRASP	209 a 348
Dias *et al.*	2002	Algoritmo Genético	102 a 251
Mauri e Lorena	2004	Algoritmo de treinamento populacional (ATP)	25 a 500
Santos	2007	geração de colunas e AG	19 a 138
Silva e Cunha	2010	GRASP + BVGP	392 a 945
Gonçalves	2010	ILS + Busca Tabu	27 a 515
De Leone et al.	2011	GRASP	80 a 400

Uma abordagem integrada do problema de escalonamento de veículos (PEV) e de motoristas (PEM) para diversas garagens também é apresentada por De Groot e Huisman (2004) e Huisman *et al.* (2005), no qual são propostos dois diferentes modelos de algoritmos baseados em uma combinação de técnicas de geração de colunas com relaxação Lagrangeana, resolvendo instâncias com 194 a 653 viagens, obtidas de dados reais de empresas de transporte público da Holanda.

Laurent e Hao (2008) resolveram um problema PEV + PEM de forma integrada e sequencial utilizando a meta-heurística GRASP com instâncias reais com até 249 viagens. A Tabela 2.3 mostra estas propostas.

Tabela 2.3: Abordagem integrada para resolução do PEV e PEM

Autor	Ano	Técnica	No. viagens
De Groot e Huisman	2004	geração de colunas + relaxação Lagrangeana	194 a 653
Huisman	2005	geração de colunas + relaxação Lagrangeana	194 a 653
Laurent e Hao	2008	GRASP	59 a 249

A utilização de métodos exatos para resolução do PEM está restrita a instâncias com dezenas ou algumas centenas de viagens, com destaque ao trabalho de Yunes *et al.* com 246 viagens.

Para instâncias de maior porte vários trabalhos utilizando métodos heurísticos são propostos para um número máximo de 945 viagens no trabalho de Silva e Cunha (2010).

Como as instâncias utilizadas neste trabalho são de grande porte, com um número de viagens entre 130 e 2313, justifica-se a utilização de um método heurístico para a resolução do PEM.

3 Revisão da Literatura

Este capítulo apresenta a descrição do problema de atribuição, técnicas e procedimentos busca local e a meta-heurística VNS, que serão utilizados na construção do algoritmo proposto.

3.1 Problema de Atribuição

O Problema de Atribuição (PA), também encontrado na literatura como Problema de Designação ou *Assignment Problem,* é um clássico problema de Otimização Combinatória em Pesquisa Operacional. De acordo com Hillier e Lieberman (2010), é um tipo especial de problema de programação linear no qual os designados são indicados para a realização de tarefas. Os exemplos mais comuns do problema de atribuição são as situações de designar tarefas para máquinas, tarefas para funcionários ou funcionários para máquinas.

A solução do PA equivale ao emparelhamento perfeito de custo mínimo em um grafo bipartido. Assim, dada uma matriz de custos de dimensões $n \times n$, o problema consiste em associar cada linha i a uma coluna j sob um custo c_{ij}, de modo a minimizar a soma dos custos.

A formulação matemática do problema é dada a seguir:

$$\text{Minimizar} \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}.x_{ij}; \tag{3.1}$$

$$\text{Sujeito a:} \quad \sum_{i=1}^{n} x_{ij} = 1; \qquad j = 1, \ldots, n; \tag{3.2}$$

$$\sum_{j=1}^{n} x_{ij} = 1; \qquad i = 1, \ldots, n; \tag{3.3}$$

$$x_{ij} \in \{0,1\}; \qquad i, j = 1, \ldots, n; \tag{3.4}$$

A atribuição de uma linha i a uma coluna j faz com que x_{ij} assuma valor 1 ou que assuma 0, caso contrário. A função-objetivo (3.1) minimiza o custo da soma das

atribuições entre linhas e colunas; a restrição (3.2) exige que para cada linha haja uma coluna associada; a restrição (3.3) garante que a cada coluna seja designada uma linha; a restrição (3.4) permite que a variável envolvida na atribuição de colunas assuma apenas os valores de decisão 0 e 1.

Em outras palavras, de acordo com Hillier e Lieberman (2010) o problema de designação deve satisfazer as seguintes hipóteses:

- O número de designados e o número de tarefas são iguais, sendo representado por n;
- Deve-se atribuir a cada designado exatamente uma tarefa;
- Cada tarefa deve ser realizada exatamente por um designado;
- Há um custo associado ao designado i (i = 1, 2,..., n) executando a tarefa j (j = 1, 2,..., n);
- O objetivo é determinar como todas as n designações devem ser feitas para minimizar o custo total.

Os problemas que não são balanceados, ou seja, aqueles para os quais não há igual número de designados e tarefas, podem ser facilmente reformulados e balanceados através da introdução de tarefas fictícias, caso o número de designados seja maior que o número de tarefas, ou de designados fictícios, em caso contrário.

Um dos algoritmos de otimização mais utilizados para a resolução desse problema é o Método Húngaro desenvolvido por Kunh (1955) na década de 50. Outro algoritmo para a resolução do problema foi proposto por Carpaneto e Toth (1987). Esse algoritmo combina o Método Húngaro com o procedimento do Menor Caminho Aumentante (*shortest augmenting path)*, garante a obtenção da solução ótima e tem complexidade $O(n^3)$. Além disso, de acordo com experimentos realizados pelos autores, o algoritmo proposto se mostrou mais rápido que o método húngaro na maioria das instâncias utilizadas.

3.2 Técnicas de Busca Local

Em problemas de minimização, dada uma solução S_0 de entrada, uma heurística de busca local deve ser mover de S_0 para um mínimo local S'. Hansen *et al.* (2010) descrevem duas técnicas para construção de heurísticas de busca local denominadas *Best improvement* e *First Improvement*.

Na heurística *Best Improvement*, parte-se de uma solução de entrada $S' = S_0$. A cada iteração S' é substituída por $S = \min\{S'' \in N(S')\}$ enquanto $f(S) < f(S')$. Se $f(S) > f(S')$, o processo termina. Esta técnica consiste em explorar toda a vizinhança e se mover para o vizinho com menor valor de função objetivo, enquanto for menor que o valor da solução corrente. A Figura 3.1 apresenta o pseudocódigo desta técnica.

```
    BestImprovement (S);
1   S ← S0;
2   repeat
3       S' ← S;
4       S ← argmin_{y∈N(S)} f(y);
5   until (f(S) ≥ f(S'));
```

Figura 3.1: Heurística *Best Improvement*

Em casos onde o *Best Improvement* consuma muito tempo de CPU, devido ao tamanho da vizinhança explorada, uma alternativa é a utilização do *First Improvement* que se baseia em mover para o primeiro vizinho que tenha o valor da função objetivo menor que o valor corrente da solução, sem explorar toda a vizinhança. Ou seja, a partir de uma solução entrada $S' = S_0$, a cada iteração, S' é substituída pela primeira solução S_i encontrada em $N(S')$ que satisfaça $f(S_i) < f(S')$. O processo é repetido enquanto existe $S_i \in N(S')\}$ tal que $f(S_i) < f(S')$, caso contrário termina. A Figura 3.2 apresenta o pseudocódigo desta técnica.

```
    FirstImprovement (S);
1   S ← S0;
2   repeat
3       S' ← S; i ← 0;
4       repeat
5           i ← i + 1;
6           S ← argmin{ f(S), f(Si)}, Si ∈ N(S) ;
7       until f(S) < f(Si) or i = |N(S)|;
8   until (f(S) ≥ f(S'));
```

Figura 3.2: Heurística *First Improvement*

3.3 Meta-heurística VNS

A meta-heurística VNS (*Variable Niegboorhood Search* ou Método de Pesquisa em Vizinhança Variável) (Hansen *et al.*, 2010) é um método de uso geral para resolução de problemas de otimização combinatória. É um método de busca local que consiste em explorar o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança com o objetivo de escapar de mínimos locais.

Diferente de outras meta-heurísticas baseadas em métodos de busca local, o método VNS não segue uma trajetória, mas sim explora vizinhanças gradativamente mais distantes da solução corrente e focaliza a busca em torno de uma nova solução se, e somente se, um movimento de melhora é realizado.

Uma versão básica do método VNS é apresentada no pseudocódigo ilustrado na Figura 3.3, onde S representa a solução corrente, z_{max} a quantidade máxima de estruturas de vizinhanças e t_{max} o tempo máximo de processamento.

```
        VNS(S, z_max);
1       z ← 1
2       repeat
3               S´ ← Shake(S,z)
4               S´´ ← Local Search(S´)
5               if f(S´´) < f(S) then
6                       S ← S´´; z ← 1;
                else
7                       z ← z + 1;
        until z = z_max;
```

Figura 3.3: Pseudocódigo VNS

Para evitar ciclos, antes de aplicar uma busca local, soluções são selecionadas aleatoriamente pelo procedimento *Shake* (linha 4), que possui como parâmetros a solução corrente e a estrutura de vizinhança selecionada. O método inclui, também, um procedimento de busca local (*Local Search* - linha 5) a ser aplicado sobre soluções vizinhas selecionadas. Esta rotina de busca local também pode usar diferentes estruturas de vizinhança, como por exemplo, aplicar a meta-heurística VND.

3.4 Procedimento PCR

O PCR (Processo de Corte e Recombinação) é um procedimento de busca local em uma estrutura de vizinhança que visa o melhoramento a partir de uma solução inicial de problemas de escalonamento.

Melo *et al*. (2010), Rizzato *et al*. (2010) e Constantino *et al*. (2009) utilizaram o PCR na fase de melhoramento de um algoritmo para resolução do problema de geração de escalas de trabalho de enfermeiros considerando as preferências dos funcionários pelos turnos. Constantino *et al*. (2011) aplicaram o procedimento PCR em um problema de escalonamento de veículos e pessoal para agroindústria, especificamente frigorífico de aves, objetivando minimizar a ociosidade da unidade de processamento. Calvi (2005) utilizou procedimento semelhante ao PCR, denominação M1 para solução do problema de escalonamento de tripulação, utilizando instâncias reais e aleatórias com até 2313 viagens. Rizzato *et al.* (2012) utilizaram o PCR para resolução do problema de escalonamento de motoristas com um algoritmo determinístico e outro em conjunto com a meta-heurística VND.

A solução inicial pode ser descrita como sendo um grafo $G(V,A)$, onde V é o conjunto de vértices que representam viagens que deverão ser alocadas aos motoristas e A o conjunto de arcos a_{ij} que indicam a possibilidade de um mesmo motorista realizar a viagem j após a viagem i. Também, os vértices estão dispostos em camadas tais que, cada camada é composta por vértices que representam viagens que podem ser realizadas como sequência de alguma tarefa na camada imediatamente anterior, e que não podem ser realizadas após tarefas localizadas na mesma camada ou camadas posteriores, conforme ilustrado na Figura 3.4.

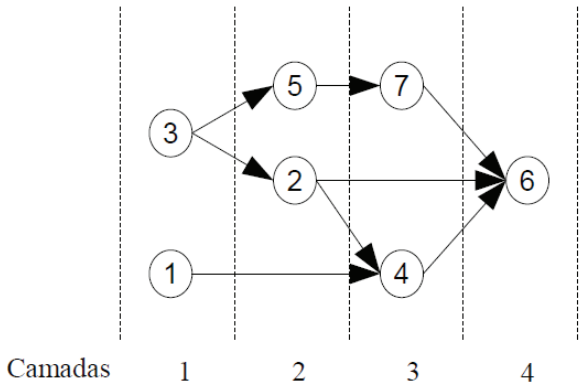

Figura 3.4: Camadas do Grafo G

Denomina-se "corte na camada x" como sendo um corte realizado entre as camadas x e x+1, isto é, divide-se a escala de jornadas em dois subconjuntos de jornadas, um à esquerda e outro à direita do corte.

Considerando uma escala com m camadas, o PCR realiza cortes em (m-1) camadas da escala de jornadas, e para cada corte, recombina os segmentos de jornadas à esquerda com segmentos de jornadas à direita do corte.

Para cada recombinação é atribuído um valor a um elemento d^x_{ij} de uma matriz D^x de ordem n, onde n é o total de jornadas da escala, x é o número da camada de corte e $d^x_{ij} = g(i,j)$ corresponde ao custo da escala de jornadas resultante da recombinação da jornada i à esquerda do corte com a jornada j à direita do corte.

O custo de cada um dos elementos da matriz é calculado conforme o seguinte critério:

- Se a recombinação for possível, o elemento da matriz receberá o custo total da escala de jornadas $g(i,j)$ para esta recombinação.
- Se a recombinação for impossível, um custo infinito será associado ao elemento da matriz. Entende-se como recombinação impossível caso ocorram as seguintes condições:
  - O local de término do segmento de jornada à esquerda do corte for diferente do local de início do segmento de jornada à direita do corte
  - O horário de término do segmento de jornada à esquerda do corte for maior que horário de início do segmento de jornada à direita do corte

A Figura 3.5 ilustra a matriz de custos D^x onde as linhas da matriz correspondem às jornadas à esquerda do corte e as colunas correspondem às jornadas à direita do corte.

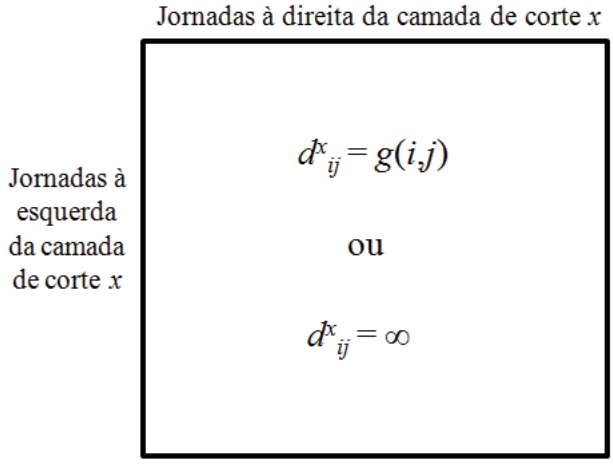

Figura 3.5: Estrutura da matriz de custos na execução do PCR

Observa-se que esta matriz é simétrica, já que o custo da recombinação do segmento de jornada *i* à esquerda do corte com o segmento de jornada *j* à direita do corte é o mesmo que o custo da recombinação do segmento de jornada *j* à esquerda do corte com o segmento de jornada *i* à direita do corte.

A matriz é resolvida por um PA que retorna as recombinações que produzem menor custo na escala de jornadas. Esta sequência de operações está ilustrada na Figura 3.6.

Procedimento PCR
Início;
Considerar uma solução *S*;
Para *x* = 1 **até** (*número de camadas* -1) **faça**
Efetuar um corte na camada *x* de *S*;
Construir a matriz de custos D^x, referente ao corte na camada *x*;
Resolver o PA da matriz D^x;
Estabelecer a solução conforme a solução do PA;
Fim.

Figura 3.6: Sequência de passos para o PCR

No exemplo a seguir será apresentado o processo de corte e recombinação na camada 3 (entre as camadas 3 e 4) na escala de jornadas a partir da solução inicial ou escala inicial de jornadas ilustrada pela Figura 3.7.

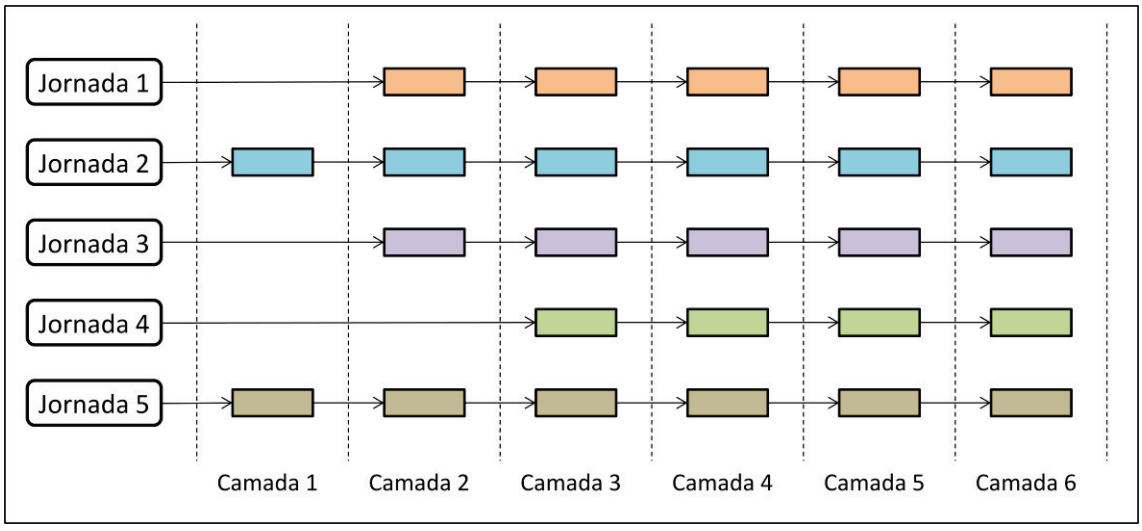

Figura 3.7: Escala inicial de jornadas

Após o corte realizado na camada 3 ilustrado na Figura 3.8, as jornadas à esquerda do corte são recombinadas com os segmentos de jornadas à direita do corte, ilustrado na Figura 3.9.

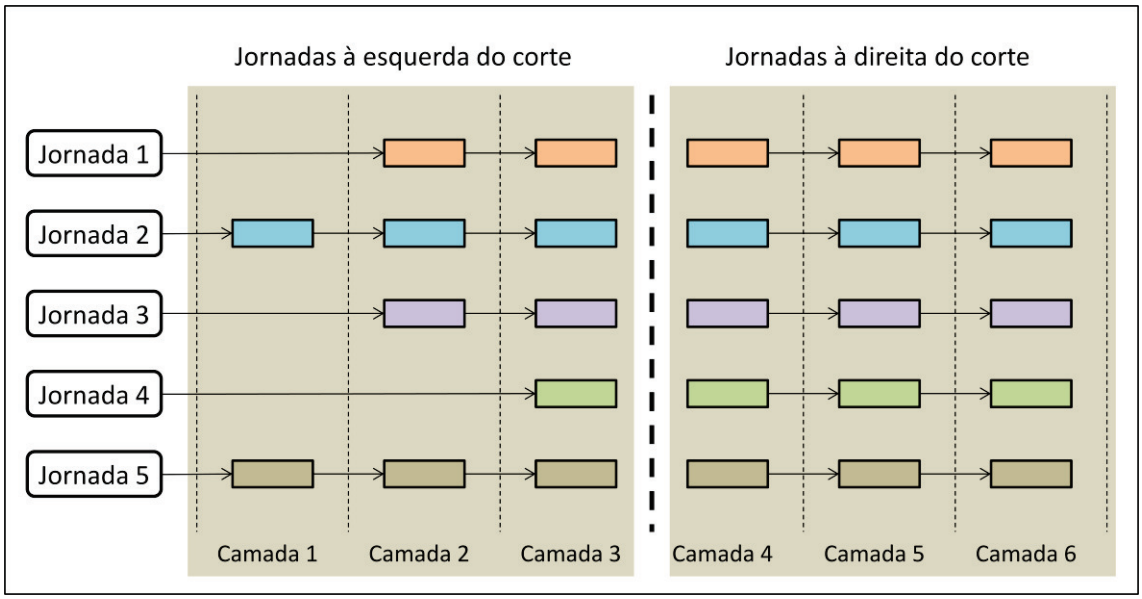

Figura 3.8: Exemplo de corte realizado na camada 3

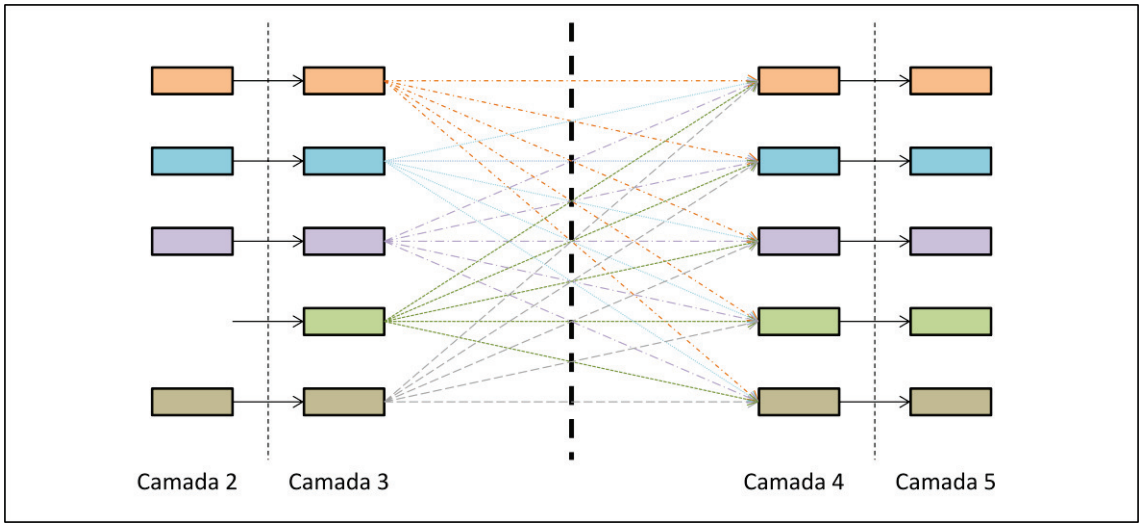

Figura 3.9: Recombinações entre jornadas na camada de corte

Monta-se a matriz custos D^3 e resolve-se o PA para D^3. A Tabela 3.1 e a Figura 3.10 ilustram um exemplo de saída gerada pelo PA e as recombinações resultantes.

Tabela 3.1: Exemplo de solução do PA

Jornada à esquerda	Jornada à direita
1	5
2	3
3	1
5	2

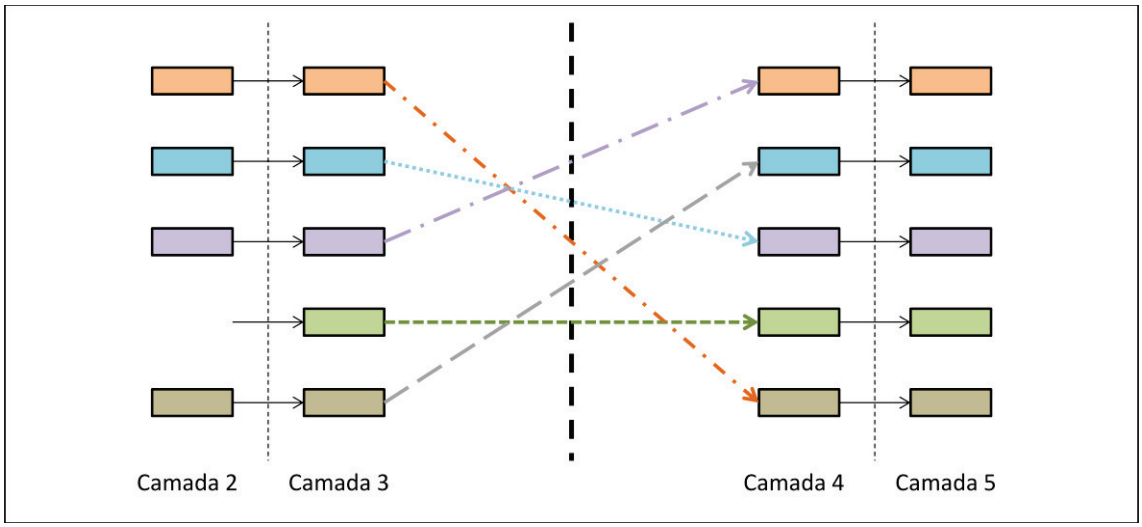

Figura 3.10: Recombinações geradas pela solução do PA

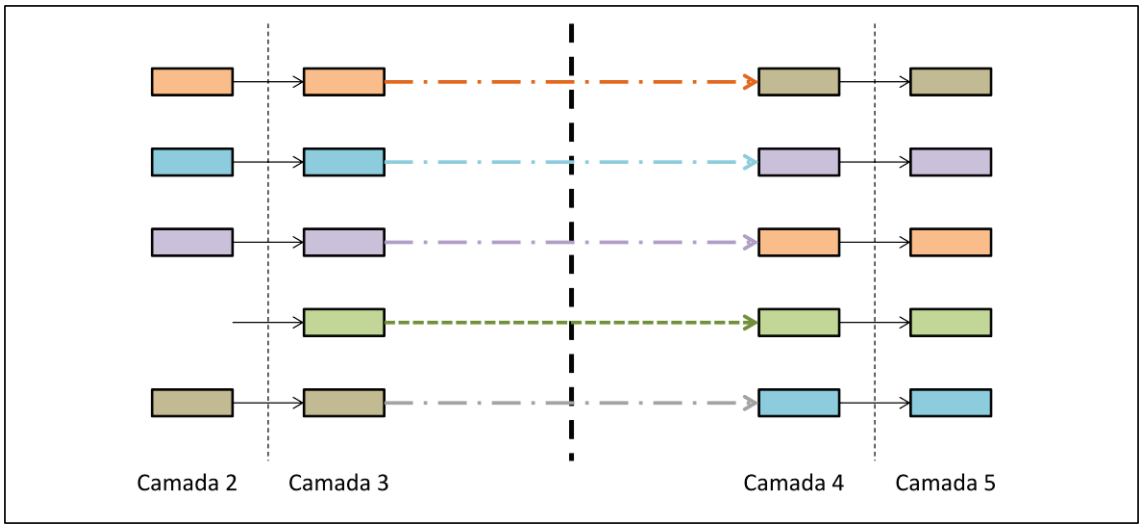

Figura 3.11: Recombinações alinhadas

A Figura 3.11 ilustra as recombinações alinhadas a cada jornada e a Figura 3.12 ilustra a escala de jornadas resultante após oprocedimento de corte e recombinações.

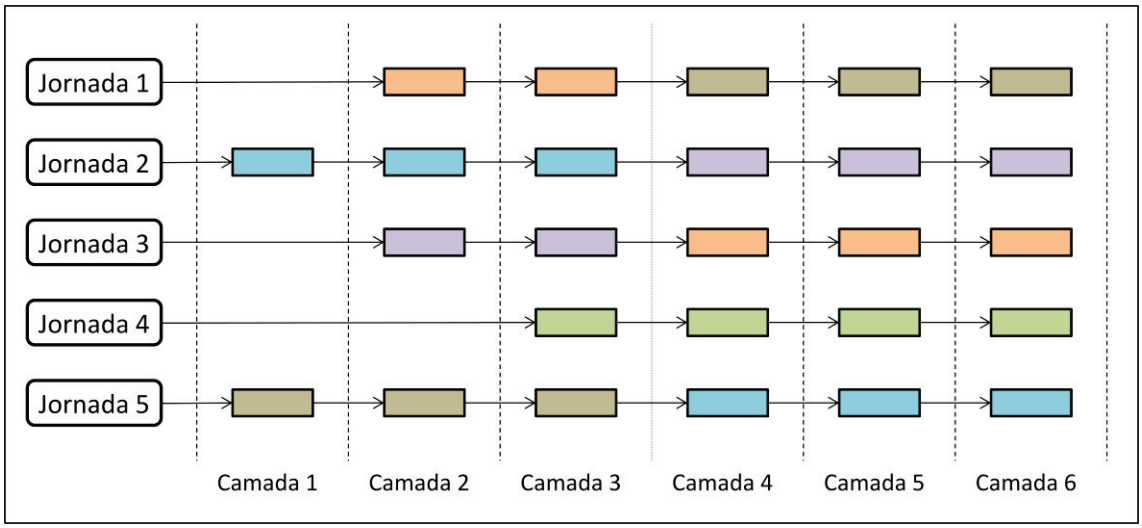

Figura 3.12: Escala de jornadas resultante

3.5 Procedimento *k-swap*

O *k-swap* é um procedimento de busca local em uma estrutura de vizinhança e opera de forma semelhante ao PCR realizando corte e recombinações. A diferença entre os procedimentos é que o PCR realiza somente um corte na escala de jornadas ao passo que o *k-swap* realiza dois cortes, o primeiro na camada x e o segundo k camadas subsequentes a camada x, ou seja, na camada $(x+k)$.

Em uma escala de jornadas com m camadas, o *k-swap* realiza $(m\text{-}1\text{-}k)$ cortes, recombina os segmentos de jornadas entre as camadas x e $(x+k)$ com os segmentos de jornadas à esquerda da camada x e à direita da camada $(x+k)$. Além disso, as camadas anteriores ao corte x e as camadas posteriores ao corte $(x+k)$ não sofrerão trocas na mesma iteração.

Para cada recombinação é atribuído um valor a um elemento e^{x}_{ij} de uma matriz E^x de ordem n, onde n é o total de jornadas da escala, x é o número da camada de corte e $e^{x}_{ij} = g(i,j)$ corresponde ao custo da escala de jornadas resultante da recombinação do segmento de jornada entre as camadas x e $(x+k)$ com a jornada i à esquerda do primeiro corte e a jornada j à direita do segundo corte.

A Figura 3.5 ilustra a matriz de custos E^x onde as linhas da matriz correspondem às jornadas à esquerda do primeiro corte e as colunas correspondem às jornadas à direita do segundo corte.

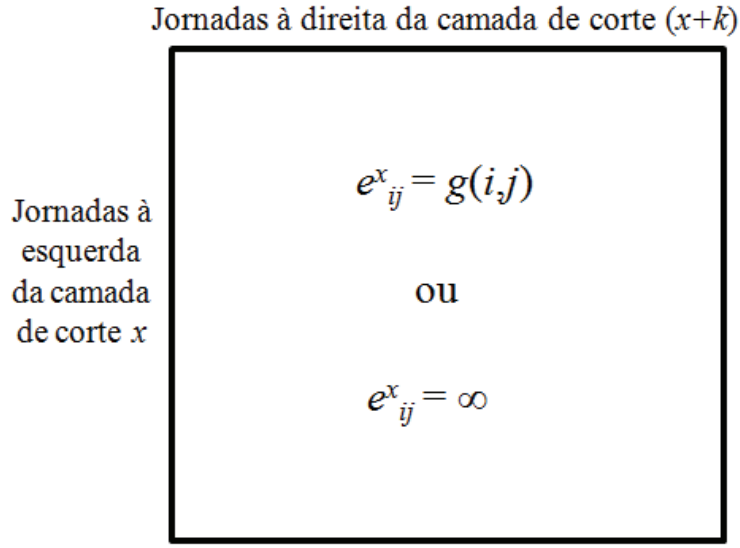

Figura 3.13: Estrutura da matriz de custos na execução do *k-swap*

A matriz é resolvida por um PA que retorna as recombinações que produzem menor custo na escala de jornadas.

A Figura 3.14 ilustra a sequência de operações para realização do procedimento *k-swap*.

Procedimento *k-swap*
Início;
Considerar uma solução *S*;
Para *x* = 1 **até** (*número de camadas* + *x* - *1*) **faça**
Efetuar um corte na camada *x* e outro na camada (*x*+*k*) de *S*;
Construir a matriz de custos E^x;
Resolver o PA da matriz E^x;
Estabelecer a solução conforme a solução do PA;
Fim.

Figura 3.14: Sequência de passos para o k-swap

Para *k* igual a 2 e um corte realizado na camada 2 é exemplificado o procedimento k-swap. A Figura 3.15 ilustra os cortes realizados na camada 2 e na camada 4 (2+2).

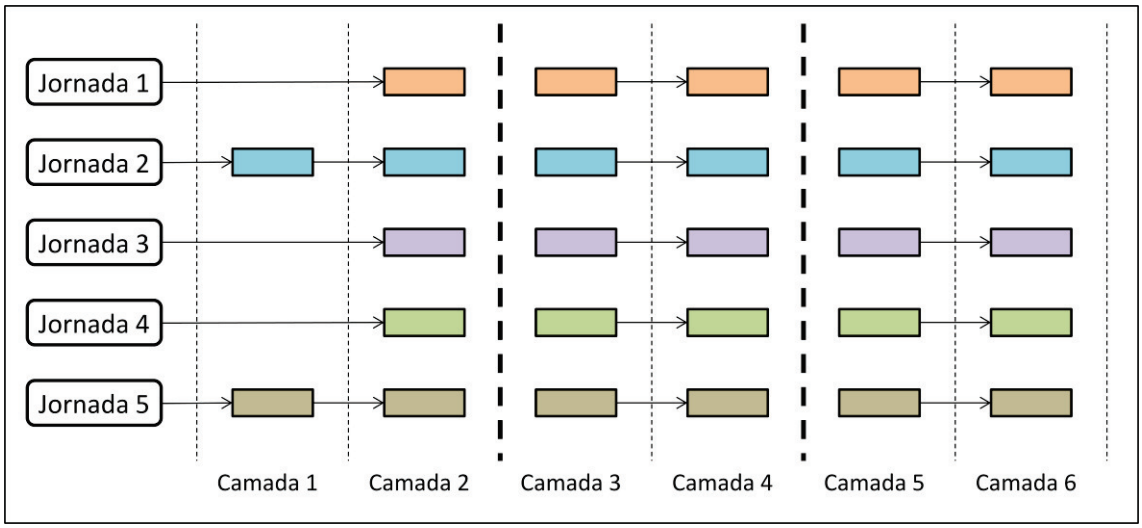

Figura 3.15: Corte nas camadas 2 e 4

Os segmentos de jornadas entre as camadas 2 e 4 são recombinados com os segmentos de jornadas a esquerda da camada 2 e à direita da camada 4, o que é ilustrado pela Figura 3.16.

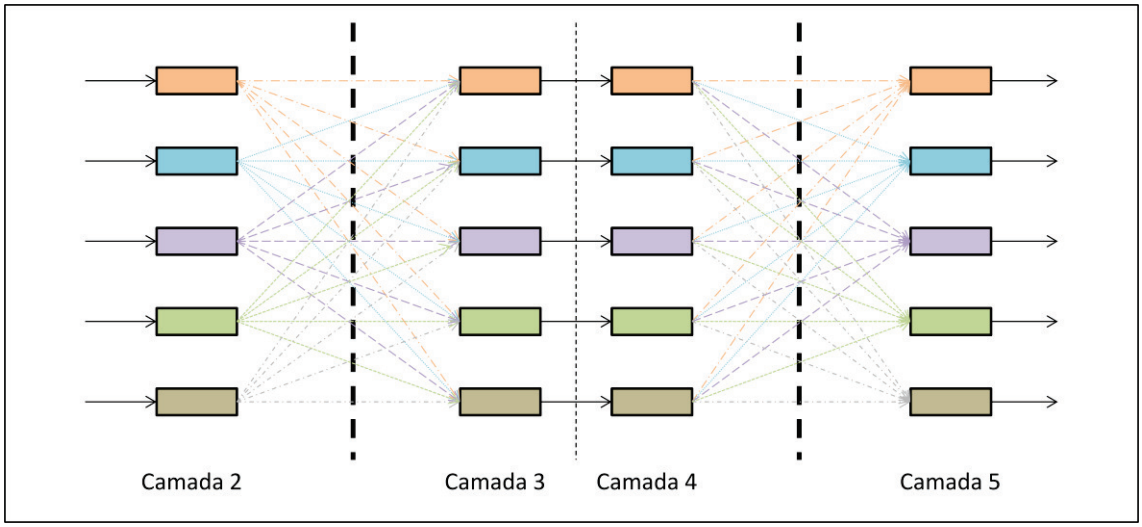

Figura 3.16: Exemplo de corte e recombinação no *k-swap*

A cada recombinação será calculado o custo da escala de jornadas e associado a um elemento da matriz E^x. A matriz é resolvida por um PA que retornará o conjunto de recombinações ilustrado pela Figura 3.17.

A Figura 3.18 ilustra as recombinações geradas pela solução do PA, de forma alinhada e a Figura 3.19 ilustra a nova escala de jornadas após a aplicação do procedimento 2-*swap* com corte realizado na camada dois.

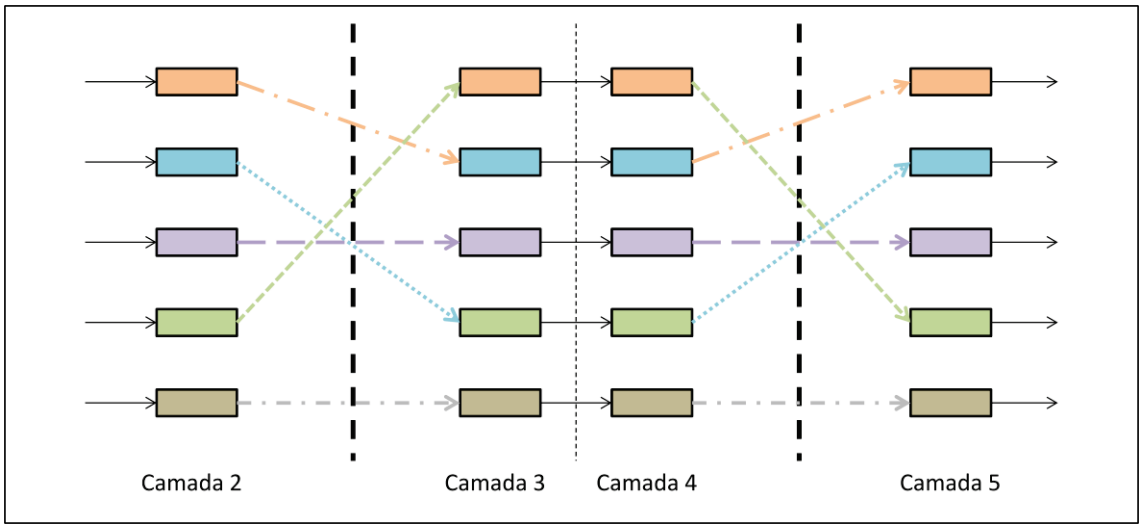

Figura 3.17: Recombinações geradas pela solução do PA

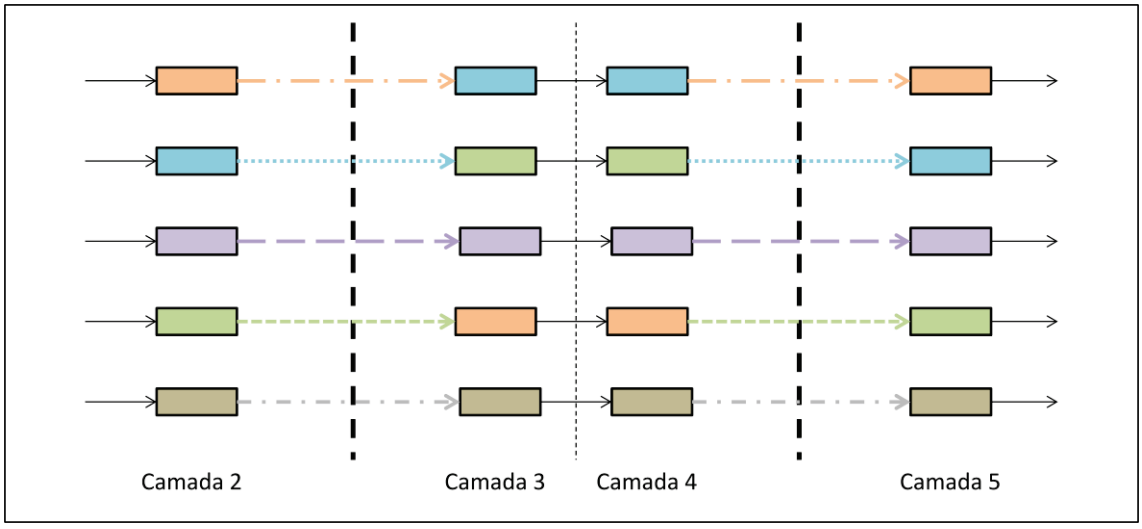

Figura 3.18: Recombinações alinhadas para o *2-swap*

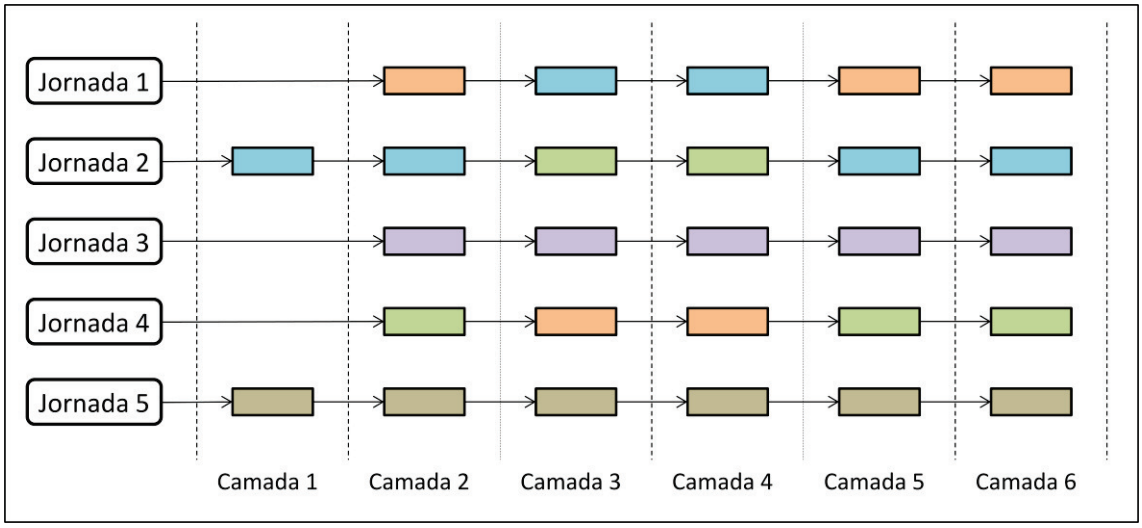

Figura 3.19: Escala de jornadas após 2-*swap* na camada dois

3.6 Gráficos *Boxplot*

Em 1977, John Tukey publicou uma proposta que posteriormente foi reconhecida como sendo um eficiente método para mostrar cinco números que sumarizam qualquer conjunto de dados. O gráfico proposto é chamado de *Boxplot* (também conhecido como *box and whisker plot*).

O *Boxplot* ou gráfico de caixas é um gráfico estatístico que possibilita representar a distribuição de um conjunto de dados com base nos seguintes parâmetros descritivos (Capela, 2011): valor mínimo, primeiro quartil, mediana ou segundo quartil, terceiro quartil e valor máximo.

A Figura 3.20 ilustra um gráfico Boxplot, onde a linha central da caixa marca a mediana do conjunto de dados e a amplitude interquartílica (AIQ) que é igual ao (3º quartil – 1º quartil). A parte inferior da caixa é delimitada pelo 1º quartil e a parte superior pelo 3º quartil.

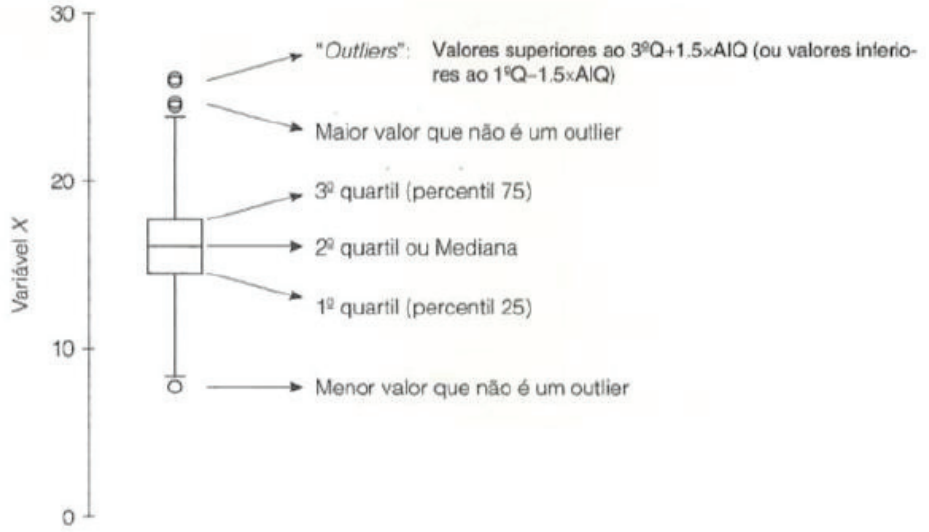

Figura 3.20: Gráfico Boxplot

As hastes inferiores e superiores se estendem, respectivamente, do quartil inferior até o menor valor não inferior a (1o quartil) – (1,5AIQ) e do quartil superior até o maior valor não superior a (3o quartil) + (1,5AIQ). As quantidades (1o quartil) – (1,5AIQ) e (3o quartil) + (1,5AIQ) delimitam, respectivamente, as cercas inferior e superior e constituem limites para além dos quais os dados passam a ser considerados *outliers*.

A existência de *outliers*, valores extremamente altos ou extremamente baixos, pode indicar tanto dados incorretos como dados válidos que carecem de uma atenção especial. Dependendo do objetivo pode ser que justamente os *outliers* sejam os pontos de interesse da análise.

O *Boxplot* permite avaliar a simetria dos dados, sua dispersão e a existência ou não de *outliers*, sendo especialmente adequado para a comparação de dois ou mais conjunto de dados correspondentes às categorias de uma variável.

4 Proposta

Este trabalho propõe duas abordagens para o PEM, sendo uma delas através de algoritmos determinísticos e outra com algoritmos utilizando a meta-heurística VNS. Todos os algoritmos propostos são baseados na utilização dos procedimentos PCR e *k-swap* na fase de melhoramento e utilizam o mesmo procedimento para a construção da solução inicial.

O texto do capítulo está organizado como segue: A seção 4.1 define a função objetivo utilizada para determinar o custo de uma escala de jornadas a qual temos como objetivo a sua minimização. A seção 4.2 apresenta detalhes do algoritmo que gera a escala de camadas a partir da escala de veículos. Este algoritmo faz parte da construção da solução inicial. A seção 4.3 apresenta o algoritmo para a construção da solução inicial a partir de uma escala de veículos que é a entrada para o PEM. A seção 4.4 apresenta os procedimentos utilizados na fase de melhoramento a partir de uma solução inicial e a seção 4.5 apresenta as instâncias utilizadas nos experimentos realizados.

4.1 Função Objetivo

A função objetivo $f : S \rightarrow R$ aplicada ao PEM associa a cada solução $x \in S$ um número real que representa o somatório do custo individual de cada jornada acrescido de penalidades, conforme a expressão:

$$f(x) = \sum_{n} \left(M_h + M_{hn} + M_{he} + N_{tr} P_{tr} + P_{JC} + P_I + P_{JM} + P_{he} + P_{me} \right)$$

onde:

M_h: minutos trabalhados e ociosos em horário normal;

M_{hn}: minutos trabalhados o ociosos em horário noturno;

M_{he}: minutos trabalhados e ociosos com hora extra;

Ntr: número de trocas efetuadas em uma jornada;

P_{tr}: penalidade em minutos por cada troca efetuada;

P_{JC}: penalidade em minutos por jornada contínua de trabalho maior que 6 (seis) horas;

P_I: penalidade em minutos por tempo de intervalo menor que 1 (uma) hora e 30 (trinta) minutos ou maior que 5 (cinco) horas;

P_{JM}: penalidade em minutos por tempo de jornada total maior que 13 (treze) horas;

P_{he}: penalidade em minutos por jornada possuir horas extras, até 2 (duas) horas;
P_{me}: penalidade em minutos por jornada ultrapassar o limite máximo de 2 (duas) horas extras.

A Tabela 4.1 ilustra os valores adotados para cada uma das penalizações utilizadas no cálculo da função objetivo.

Tabela 4.1: Tabela de penalizações

Sigla	Penalidade	Valor
P_{tr}	penalidade em minutos por cada troca efetuada	110
P_{JC}	penalidade em minutos por jornada contínua de trabalho maior que 6 (seis) horas	7000
P_I:	penalidade em minutos por tempo de intervalo menor que 1 (uma) hora e 30 (trinta) minutos ou maior que 5 (cinco) horas	7000
P_{JM}	penalidade em minutos por tempo de jornada total maior que 13 (treze) horas	7000
P_{he}	penalidade em minutos por jornada possuir horas extras, até 2 (duas) horas	100
P_{me}	penalidade em minutos por jornada ultrapassar o limite máximo de 2 (duas) horas extras	5000

4.2 Geração de Camadas

Para que o problema possa ser tratado, a escala de veículos deve ser transformada em uma escala de camadas para posterior atribuição das viagens às jornadas.

A distribuição das viagens na escala de camadas deve ser feita observando-se o seguinte critério: as viagens pertencentes a uma determinada camada podem ser realizadas como sequência de alguma viagem pertencente a uma camada imediatamente anterior e que não podem ser realizadas após viagens pertencentes a mesma camada ou camadas posteriores.

A Figura 4.1 ilustra o pseudocódigo do algoritmo para gerar camadas considerando as seguintes definições:

- Seja T o conjunto de todas as viagens na escala;
- Seja $Seq(TP, T_k)$ uma função que retorna *verdadeiro* se uma viagem TP pode ser sequenciada com alguma viagem da camada T_k e retorna *falso* em caso contrário. Esta função deve levar em consideração, restrições operacionais como os tempos e locais de término e início das viagens envolvidas;

- Seja *MaisCedo*(*T*) uma função que devolve a viagem que inicia mais cedo em *T*;
- Seja *CriarCamada* uma função que cria uma nova camada vazia.

```
Algoritmo GeraCamadas

Begin
1   k=1;
2   T_k = CriarCamada; //Cria a primeira camada
3   while T≠ø (vazio) do
4           TP= MaisCedo(T);
5           T=T-{TP};
5           i=0;
7           repeat  //procura por uma camada existente para inserir TP.
8                   i=i+1;
9           until (i>k)  or  not Seq(TP, T_i)
10          if  i>k  then
11                  k=i;  //nova camada é criada
12                  T_k = CriarCamada;
13          T_i=T_i + {TP};
End;
```

Figura 4.1: Algoritmo Gera Camadas

Para a descrição dos procedimentos PCR e *k-swap* nas seções 3.4 e 3.5, foi utilizada uma notação onde as camadas em uma escala de jornadas eram representadas por retas verticais. Na realidade a divisão de camadas em uma escala de tempo não seria desta forma.

Seja um bloco de viagens pertencentes a seis linhas de uma escala de veículos, retiradas de uma instância real, contendo somente viagens do início da escala, mostrada na Tabela 4.2.

A coluna Linha (coluna 1) identifica o número da linha, da escala de veículos, numeradas de 1 à 6. Cada linha da tabela representa uma viagem que o veículo executa entre duas oportunidades de troca, representadas pelas colunas Origem (coluna 2) e Destino (coluna 3). Nestas colunas, a letra “G” indica garagem e a letra “T” indica terminal de passageiros. Os horários de início e término de cada viagem assim como a sua duração também são mostrados na tabela, respectivamente nas colunas Início (coluna 4), Término (coluna 5) e Duração (coluna 6).

Os valores representados nas colunas 4, 5 e 6 estão no formato HH(hora):MM(minuto).

Tabela 4.2: Exemplo de escala para seis veículos

Linha	Origem	Destino	Início	Término	Duração
1	G	T	05:25	06:10	00:45
1	T	T	06:10	07:00	00:50
1	T	T	07:00	08:00	01:00
1	T	T	08:00	09:10	01:10
1	T	T	09:10	10:30	01:20
2	G	T	04:40	05:30	00:50
2	T	T	05:30	06:30	01:00
2	T	T	06:30	07:30	01:00
2	T	T	07:30	08:30	01:00
2	T	T	08:30	09:30	01:00
3	G	T	06:00	07:05	01:05
3	T	T	07:05	08:35	01:30
3	T	T	08:35	10:05	01:30
4	G	T	05:40	06:35	00:55
4	T	T	06:35	07:40	01:05
4	T	T	07:40	08:40	01:00
4	T	T	08:40	09:35	00:55
5	G	T	05:30	06:20	00:50
5	T	T	06:20	07:20	01:00
5	T	T	07:20	08:20	01:00
5	T	T	08:20	09:20	01:00
6	G	T	05:00	05:30	00:30
6	T	T	05:30	07:00	01:30
6	T	T	07:00	08:30	01:30
6	T	T	08:30	10:00	01:30

A Figura 4.2(a) mostra a mesma escala de veículos em um gráfico de tempo. Os blocos com a mesma cor representam viagens de uma mesma linha.

As letras G e T nas extremidades dos blocos indicam os locais onde temos oportunidades de trocas, isto é, G (garagem) e T (terminal). Também está indicado no interior de cada bloco o horário de início e de término da viagem correspondente.

Aplicando o algoritmo gerador de camadas, teremos uma escala de camadas conforme ilustrado pela Figura 4.2(b), onde os blocos de mesma cor representam as viagens pertencentes a mesma camada.

Observa-se que as viagens que compõem uma camada não são separadas por linhas retas. Desta forma será adotada a notação utilizada nas seções 3.4 e 3.5 para representação das viagens dentro de uma escala de jornadas.

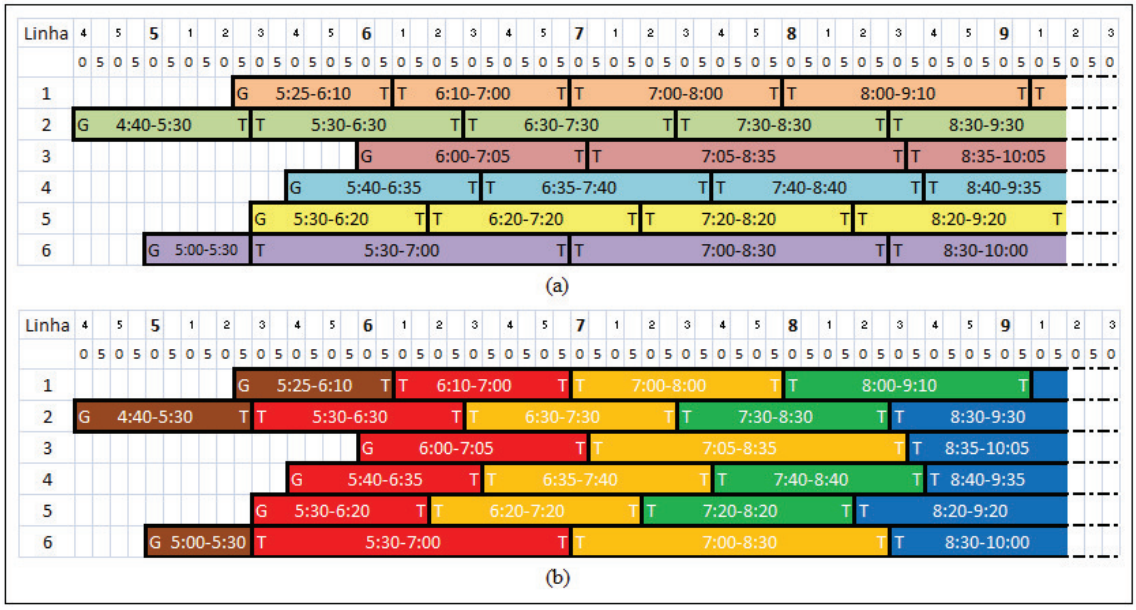

Figura 4.2: Escala de viagens e formação de camadas

A Tabela 4.3 identifica as camadas geradas pelo seu número na coluna Camada (coluna 1), a quantidade de viagens pertencente a esta camada (coluna 2) e a cor utilizada para sua ilustração (coluna 3).

Tabela 4.3: Identificação das camadas geradas

Camada	Quantidade de viagens	Cor
1	3	marrom
2	6	vermelho
3	6	amarelo
4	4	verde
5	6	azul

A camada 1 é composta por três viagens e é representada pela cor marrom. As camadas 2, 3 e 5 são compostas por seis viagens e são representadas pelas cores vermelho, amarelo e azul, respectivamente. A camada 4 possui quatro viagens e é representada pela cor verde.

A Tabela 4.4 mostra as viagens da escala de veículos da Tabela 4.2 distribuídas em camadas criadas pelo algoritmo gerador de camadas.

Tabela 4.4: Geração de camadas

Camada	Linha	Origem	Destino	Início	Término	Duração
1	1	G	T	05:25	06:10	00:45
	2	G	T	04:40	05:30	00:50
	6	G	T	05:00	05:30	00:30
2	1	T	T	06:10	07:00	00:50
	2	T	T	05:30	06:30	01:00
	3	G	T	06:00	07:05	01:05
	4	G	T	05:40	06:35	00:55
	5	G	T	05:30	06:20	00:50
	6	T	T	05:30	07:00	01:30
3	1	T	T	07:00	08:00	01:00
	2	T	T	06:30	07:30	01:00
	3	T	T	07:05	08:35	01:30
	4	T	T	06:35	07:40	01:05
	5	T	T	06:20	07:20	01:00
		T	T	07:00	08:30	01:30
4	1	T	T	08:00	09:10	01:10
	2	T	T	07:30	08:30	01:00
	4	T	T	07:40	08:40	01:00
	5	T	T	07:20	08:20	01:00
5	1	T	T	09:10	10:30	01:20
	2	T	T	08:30	09:30	01:00
	3	T	T	08:35	10:05	01:30
	4	T	T	08:40	09:35	00:55
	5	T	T	08:20	09:20	01:00
	6	T	T	08:30	10:00	01:30

4.3 Construção da Solução Inicial

Obtida a escala de camadas, a construção da solução inicial consiste na alocação das viagens de cada uma das camadas para uma determinada jornada da escala.

Esta alocação é feita através da resolução de um problema de atribuição (PA) para cada camada. O problema de atribuição para cada camada é representado pela matriz quadrada $C = [c_{ij}]$, de ordem $|V_k| + |J|$, onde V_k é o conjunto de viagens da camada k e J é o conjunto de jornadas.

Para a primeira camada, a ordem da matriz será V_k, já que o número de jornadas é zero. Para as camadas subsequentes, a ordem da matriz será $|V_k| + |J|$.

A matriz $C = [c_{ij}]$ é dividida em quatro blocos conforme ilustra a Figura 4.3, sendo seus elementos definidos da seguinte forma:

	$j = 1,..., \|V_k\|$ Viagens	$j = \|V_k\|+1, ..., \|V_k\|+\|J\|$ Viagens fictícias
Jornadas $i = 1,..., \|J\|$	**Bloco 1** $c_{ij} = c + g(i,j)$ ou $c_{ij} = \infty$	**Bloco 2** $c_{ij} = c_{JMC} - c_i + P_{to}$ ou $c_{ij} = c_i + P_{to2}$
Jornadas fictícias $i = \|J\|+1, ..., \|V_k\|+\|J\|$	**Bloco 3** $c_{ij} = c_{nj}$	**Bloco 4** $c_{ij} = 0$

Figura 4.3: Blocos da Matriz

- Bloco 1: Para i = 1,..,$|J|$ e j = 1, .. , $|V_k|$, se for possível alocar a viagem j à jornada i, temos:

$$c_{ij} = c_i + g(i,j),$$

onde c_i representa o custo em minutos pagos acumulado pela jornada i.

A função g define o custo de alocar a viagem j à jornada i e é representada por:

$$g(i,j) = M_j + M_{ji} + P_{tr} + P_{JC} + P_{JM} + P_{he} + P_{me}$$

onde:

M_j: minutos da viagem j

M_{ji}: minutos ociosos entre o final da última viagem executada pela jornada i até o início da viagem j

P_{tr}: penalidade em minutos se houver troca de veículo

P_{JC}: penalidade em minutos por jornada contínua de trabalho maior que 6 (seis) horas

P_{JM}: penalidade em minutos por tempo de jornada total maior que 13 (treze) horas

P_{he}: penalidade em minutos por jornada possuir até 2 (duas) horas extras

P_{me}: penalidade em minutos por jornada ultrapassar o limite máximo de 2 (duas) horas extras

Se não for possível alocar a viagem j à jornada i, $c_{ij} = \infty$. A impossibilidade de alocar a viagem j à jornada i é determinada por:

- horário de término da última viagem da jornada i após início da viagem j;
- local de término da última viagem da jornada i diferente do local de início da viagem j.

- Bloco 2: Para $i = 1,..,|J|$ e $j = |V_k|+1, .. , |V_k|+|J|$, é o custo atribuído a jornada i sem receber uma tarefa adicional.

  Para jornadas que ainda não tiveram folgas, terão preferência de folga as jornadas com maior tempo, acrescido de uma penalidade em minutos P_{to}, conforme a equação:

$$c_{ij} = c_{JMC} - c_i + P_{to}$$

  onde c_{JMC} representa o custo máximo de uma jornada contínua e c_i representa o custo em minutos pagos acumulado pela jornada i.

  Para jornadas que já tiveram folga, uma penalidade maior (P_{to2}) será adicionada ao custo para evitar as jornadas com mais de um intervalo. O custo para este caso pode ser representado por:

$$c_{ij} = c_i + P_{to2}$$

- Bloco 3: Para $i = |J|+1, .. ,|J|+|V_k|$ e j = 1, .. , $|V_k|$, contém jornadas fictícias e uma viagem j é alocada para uma jornada i se esta viagem não puder ser alocada para nenhuma das jornadas existentes. Recebe o custo de uma jornada de 7 (sete) horas e 20 (vinte) minutos representado por:

$$c_{ij} = c_{nj}$$

  onde c_{nj} é o custo de uma nova jornada.

- Bloco 4: Para $i = |J|+1, .. ,|J|+|V_k|$ e $j = |V_k|+1, .. , |V_k|+|J|$, o custo atribuído será zero, pois associa as jornadas fictícias com viagens fictícias, uma vez que apenas complementam a matriz.

A matriz $C = [c_{ij}]$ é resolvida por um PA e o algoritmo utilizado na implementação foi desenvolvido por Carpaneto e Toth (1987) que garante a obtenção da solução ótima e tem complexidade $O(n^3)$, descrito na seção 3.1.

A Tabela 4.5 mostra os valores utilizados para as penalidades na construção da solução inicial.

Tabela 4.5: Valores adotados para as penalidades na construção da solução inicial

Sigla	Penalidade	Valor
P_{tr}	penalidade em minutos se houver troca de veículo	110
P_{JC}	penalidade em minutos por jornada contínua de trabalho maior que 6 (seis) horas	7000
P_{JM}	penalidade em minutos por tempo de jornada total maior que 13 (treze) horas	9000
P_{he}	penalidade em minutos por jornada possuir até 2 (duas) horas extras	100
P_{me}	penalidade em minutos por jornada ultrapassar o limite máximo de 2 (duas) horas extras	5000
P_{to}	penalidade em minutos por não atribuir viagem a jornada	500
P_{to2}	penalidade em minutos por não atribuir viagem a jornada que já teve intervalo para descanso	1000

A Figura 4.4 ilustra a sequência de operações para a construção da solução inicial.

Construção da Solução Inicial

Início
1 *GeraCamadas;* //total de m camadas
2 **Para** $k = 1$ **até** m **faça**:
3 Construa a matriz de custos C para a camada k;
4 Resolva o PA tendo como entrada a matriz C;
5 Com o resultado obtido, atribua as viagens da camada k às respectivas jornadas;
Fim;

Figura 4.4: Construção da Solução Inicial

4.4 Fase de Melhoramento

Na fase de melhoramento foram implementadas duas famílias de algoritmos que utilizam os procedimentos PCR e *k-swap*, uma composta de algoritmos determinísticos e a outra de algoritmos que utilizam a meta-heurística VNS. Para o *k-swap*, foram adotados os seguintes valores para k: 1, 2, 3 e 4, conforme ilustra a Tabela 4.6.

.

Tabela 4.6: Implementações do *k-swap*

Valor de *k*	Procedimento
1	1-*swap*
2	2-*swap*
3	3-*swap*
4	4-*swap*

4.4.1 Algoritmos Determinísticos

Na abordagem determinística, a partir de uma solução inicial, cinco procedimentos de busca local são utilizados de modo sequencial até que o processo se estabilize, ou seja, quando não há melhora na solução encontrada. A Figura 4.5 ilustra o pseudocódigo desta abordagem.

```
    Determinístico(S);
Início
1   iteração ← 0;
2   repita
3           S ← S0;
4           S1 ← Procedimento1(S);
5           S2 ← Procedimento2(S1);
6           S3 ← Procedimento3(S2);
7           S4 ← Procedimento4(S3);
8           S5 ← Procedimento5(S4);
9           S0 ← S5;
10          iteração ← iteração + 1;
11  até (f(S) ≥ f(S5));
fim
```

Figura 4.5: Pseudocódigo da abordagem determinística

Os procedimentos (1, 2, 3, 4 e 5) de busca local utilizados foram o PCR, 1-*swap*, 2-*swap*, 3-*swap* e 4-*swap* e a Tabela 4.7 mostra as sequências implementadas com suas respectivas notações:

Tabela 4.7: Sequência de Procedimentos de Busca Local

Notação	**Proced. 1**	**Proced. 2**	**Proced. 3**	**Proced. 4**	**Proced. 5**
P1234	PCR	1-*swap*	2-*swap*	3-*swap*	4-*swap*
1P234	1-*swap*	PCR	2-*swap*	3-*swap*	4-*swap*
12P34	1-*swap*	2-*swap*	PCR	3-*swap*	4-*swap*
123P4	1-*swap*	2-*swap*	3-*swap*	PCR	4-*swap*
1234P	1-*swap*	2-*swap*	3-*swap*	4-*swap*	PCR
P4321	PCR	4-*swap*	3-*swap*	2-*swap*	1-*swap*
4P321	4-*swap*	PCR	3-*swap*	2-*swap*	1-*swap*
43P21	4-*swap*	3-*swap*	PCR	2-*swap*	1-*swap*
432P1	4-*swap*	3-*swap*	2-*swap*	PCR	1-*swap*
4321P	4-*swap*	3-*swap*	2-*swap*	1-*swap*	PCR
P1324	PCR	1-*swap*	3-*swap*	2-*swap*	4-*swap*
1P324	1-*swap*	PCR	3-*swap*	2-*swap*	4-*swap*
13P24	1-*swap*	3-*swap*	PCR	2-*swap*	4-*swap*
132P4	1-*swap*	3-*swap*	2-*swap*	PCR	4-*swap*
1324P	1-*swap*	3-*swap*	2-*swap*	4-*swap*	PCR
P2413	PCR	2-*swap*	4-*swap*	1-*swap*	3-*swap*
2P413	2-*swap*	PCR	4-*swap*	1-*swap*	3-*swap*
24P13	2-*swap*	4-*swap*	PCR	1-*swap*	3-*swap*
241P3	2-*swap*	4-*swap*	1-*swap*	PCR	3-*swap*
2413P	2-*swap*	4-*swap*	1-*swap*	3-*swap*	PCR

Os procedimentos de busca local PCR e *k-swap* foram implementados com as seguintes variações:

- Quanto à sequência de cortes realizados:
    - *Forward*: da primeira à camada n-1 para o PCR e da primeira à camada (n-k+1) para o *k-swap*

- *Backward*: da camada n-1 à primeira camada para o PCR e da camada (n-k+1) para a primeira camada no *k-swap*.

- Quanto às técnicas de busca local discutidas na seção 3.2:
  - *First Improvement*: a cada corte realizado, caso a solução obtida através da resolução do PA for melhor que a solução corrente, adota-se a solução obtida como a solução corrente e prossegue os cortes a serem realizados no procedimento.
  - *Best improvement*: a cada corte realizado, armazena-se cada solução obtida através da resolução do PA e após a realização de todos os cortes, compara a melhor solução com a solução corrente e caso esta seja melhor, adota-se a solução obtida como a solução corrente para a próxima etapa.

A Figura 4.6 ilustra a sequência de passos para a execução do procedimento PCR *First Improvement*/*Forward* e a Figura 4.7 ilustra a sequência de passos para a execução do procedimento PCR *Best Improvement*/Forward.

	Procedimento PCR *First Improvement/Forward*
	Início;
1	Considerar uma solução corrente *S*;
2	**Para** x = 1 **até** (*número de camadas* -1) **faça**
3	Efetuar um corte na camada x de *S*;
4	Construir a matriz de custos D^x, referente ao corte na camada x;
5	Resolver o PA da matriz D^x;
6	Se a solução obtida com a resolução do PA for melhor que S, adotá-la como a solução corrente para a próxima camada;
	Fim.

Figura 4.6: Passos para PCR First Improvement/Forward

Para a versão Backward, dos procedimentos PCR *First Improvement* e PCR *Best Improvement*, substitui-se a linha 2 por:

"Para x = (*número de camadas* – 1) **até** 1 **faça"**

Procedimento PCR *Best Improvement/Forward*

Início;

1 Considerar uma solução corrente *S*;

2 **Para** *x* = 1 **até** (*número de camadas* -1) **faça**

3 Efetuar um corte na camada *x* de *S*;

4 Construir a matriz de custos D^x, referente ao corte na camada *x*;

5 Resolver o PA da matriz D^x;

6 Armazenar a solução S^x obtida com a resolução do PA;

7 Se a melhor solução do conjunto das soluções S^x armazenadas for melhor que S, adotá-la como a solução final do procedimento;

Fim.

Figura 4.7: Passos para o PCR Best Improvement/Forward

A Figura 4.8 ilustra a sequência de passos para a execução do procedimento *k-swap First Improvement*/*Forward* e a Figura 4.9 ilustra a sequência de passos para a execução do procedimento *k-swap Best Improvement*/*Forward.*

Procedimento *k-swap First Improvement/Forward*

Início;

1 Considerar uma solução corrente *S*;

2 **Para** *x* = 1 **até** (*número de camadas* - *k* - 1) **faça**

3 Efetuar corte nas camadas *x* e (*x*+*k*) de *S*;

4 Construir a matriz de custos E^x;

5 Resolver o PA da matriz E^x;

6 Se a solução obtida com a resolução do PA for melhor que S, adotá-la como a solução corrente para a próxima camada;

Fim.

Figura 4.8: Passos para o *k-swap First Improvement*/*Forward*

Para a versão *Backward*, dos procedimentos *k-swap First Improvement* e *k-swap Best Improvement*, substitui-se a linha 2 por:

"Para *x* = (*número de camadas* - *k* - 1) **até** 1 **faça"**

Procedimento *k-swap Best Improvement/Forward*

Início;

1 Considerar uma solução corrente *S*;

2 **Para** $x = 1$ **até** (*número de camadas* - *k* -1) **faça**

3 Efetuar nas camadas x e $(x+k)$ de *S*;

4 Construir a matriz de custos E^x;

5 Resolver o PA da matriz E^x;

6 Armazenar a solução S^x obtida com a resolução do PA;

7 Se a melhor solução do conjunto das soluções S^x armazenadas for melhor que S, adotá-la como a solução final do procedimento;

Fim.

Figura 4.9: Passos para o *k-swap Best Improvement/Forward*

A Tabela 4.8 ilustra as quatro versões utilizadas para os procedimentos de busca local PCR e *k-swap* e a notação utilizada para cada combinação:

Tabela 4.8: Versões para o PCR e *k-swap*

Notação	Técnica de busca local	Sequência de cortes
FI-FW	*First Improvement*	*Forward*
BI-FW	*Best Improvement*	*Forward*
FI-BW	*First Improvement*	*Backward*
BI-BW	*Best Improvement*	*Backward*

Portanto, nesta abordagem foram implementadas 80 versões de algoritmos determinísticos, variando-se a sequência dos procedimentos, técnica de busca local e sequência de cortes. A Tabela 4.10 mostra todas as versões da abordagem determinística.

A Tabela 4.9 exemplifica com detalhes a interpretação da versão 1234P-FI-BW.

Tabela 4.9: Detalhamento da notação para versão determinística

Versão	12P34P-FI-BW
Sequência de procedimentos	1-*swap* → 2-*swap* → PCR → 3-*swap* → 4-*swap*
Técnica de busca local	*First Improvement*
Sequência de cortes	*Backward*

Tabela 4.10: Versões da abordagem determinística

P1234-FI-FW	P1234-FI-BW	P1234-BI-FW	P1234-BI-BW
1P234-FI-FW	1P234-FI-BW	1P234-BI-FW	1P234-BI-BW
12P34-FI-FW	12P34-FI-BW	12P34-BI-FW	12P34-BI-BW
123P4-FI-FW	123P4-FI-BW	123P4-BI-FW	123P4-BI-BW
1234P-FI-FW	1234P-FI-BW	1234P-BI-FW	1234P-BI-BW
P4321-FI-FW	P4321-FI-BW	P4321-BI-FW	P4321-BI-BW
4P321-FI-FW	4P321-FI-BW	4P321-BI-FW	4P321-BI-BW
43P21-FI-FW	43P21-FI-BW	43P21-BI-FW	43P21-BI-BW
432P1-FI-FW	432P1-FI-BW	432P1-BI-FW	432P1-BI-BW
4321P-FI-FW	4321P-FI-BW	4321P-BI-FW	4321P-BI-BW
P1324-FI-FW	P1324-FI-BW	P1324-BI-FW	P1324-BI-BW
1P324-FI-FW	1P324-FI-BW	1P324-BI-FW	1P324-BI-BW
13P24-FI-FW	13P24-FI-BW	13P24-BI-FW	13P24-BI-BW
132P4-FI-FW	132P4-FI-BW	132P4-BI-FW	132P4-BI-BW
1324P-FI-FW	1324P-FI-BW	1324P-BI-FW	1324P-BI-BW
P2413-FI-FW	P2413-FI-BW	P2413-BI-FW	P2413-BI-BW
2P413-FI-FW	2P413-FI-BW	2P413-BI-FW	2P413-BI-BW
24P13-FI-FW	24P13-FI-BW	24P13-BI-FW	24P13-BI-BW
241P3-FI-FW	241P3-FI-BW	241P3-BI-FW	241P3-BI-BW
2413P-FI-FW	2413P-FI-BW	2413P-BI-FW	2413P-BI-BW

4.4.2 Utilização do VNS

Para os algoritmos que utilizam a meta-heurística VNS foram implementadas três versões, onde realizamos variações nos procedimentos *Shake* e *Local Search*, descritos na seção 3.3. Em ambos os procedimentos são utilizados o PCR e o *k-swap*.

O procedimento *Shake* consiste em executar um movimento de modo aleatório para encontrar um vizinho da solução corrente e estabelecer uma nova estrutura de vizinhança.

A partir da escala de jornadas da solução corrente é selecionada aleatoriamente uma camada onde é realizado um corte. Nesta camada de corte, recombina-se as

jornadas à esquerda do corte com jornadas à direita do corte, também selecionadas aleatoriamente, em 10% das jornadas da escala. Esta é a parte comum entre as duas abordagens implementadas.

Na primeira versão (1234P), aplica-se um dos cinco procedimentos entre: 1-*swap*-BI-BW, 2-*swap*-BI-BW, 3-*swap*-BI-BW, 4-*swap*-BI-BW e PCR-BI-BW, selecionados através da variável *z*. A Tabela 4.11 mostra os valores dos procedimentos utilizados dependendo do valor de *z* e a Figura 4.10 ilustra o pseudocódigo para o procedimento *Shake* considerando *S* a solução corrente e *Rd*(*S*) uma função que seleciona aleatoriamente uma camada na escala de jornadas e realiza recombinações em 10% das jornadas selecionadas também aleatoriamente.

Tabela 4.11: Relação entre valores de *z* para *Shake* 1234P

Valor de *z*	**Procedimento a ser aplicado**
1	1-*swap*-BI-BW
2	2-*swap*-BI-BW
3	3-*swap*-BI-BW
4	4-*swap*-BI-BW
5	PCR-BI-BW

```
        Shake(S)
        Início
2       S′ = Rd(S);
3       z ← 1;
5       repita
6               caso <z> igual a
7               1: S′ ← 1-swap-BI-BW(S);
8               2: S′ ← 2-swap-BI-BW(S′);
9               3: S′ ← 3-swap-BI-BW(S′);
10              4: S′ ← 4-swap-BI-BW(S′);
11              5: S′ ← PCR-BI-BW(S′);
                fimcaso
12              se (f(S′) < f(S)) então
13                      S ← S′;
14                      z ← 1;
                senão
16                      z ← z + 1;
                fimse
        até (z>5)
        fim
```

Figura 4.10: Pseudocódigo para *Shake* 1234P

Na segunda versão de *Shake* (1P), utilizamos somente 1-*swap*-BI-BW e PCR-BI-BW selecionados pela variável *z*, conforme mostra a Tabela 4.12 e a Figura 4.11.

```
        Shake(S)
1       Início
2       S′ = Rd(S);
3       z ← 1;
4       repita
5               caso (z) igual a
6               1: S′ ← 1-swap-BI-BW(S);
7               2: S′ ← PCR-BI-BW(S′);
                fimcaso
8               se (f(S′) < f(S)) então
9                       S ← S′;
10                      z ← 1;
11              senão
12                      z ← z + 1;
                fimse
13      até (z > 2)
        fim
```

Figura 4.11: Pseudocódigo para *Shake* 1P

Considera-se *S* a solução corrente e *Rd*(*S*) uma função que seleciona aleatoriamente uma camada na escala de jornadas e realiza recombinações em 10% das jornadas selecionadas também aleatoriamente.

Tabela 4.12: Relação entre valores de *z* para *Shake* 1P

Valor de *z*	Procedimento a ser aplicado
1	1-*swap*-BI-BW
2	PCR-BI-BW

O procedimento *Local Search* foi implementado com duas versões, sendo uma delas executando a sequência 1234P-FI-BW utilizada na versão determinística e a outra utilizando somente os procedimentos 1-*swap* e PCR, *First Improvement* e *Backward*, denotados por 1P-FI-BW.

As três versões implementadas utilizando a meta-heurística VNS são mostradas na

Tabela **4.13**. A coluna 1 mostra a notação utilizada, a coluna 2 mostra o procedimento de Shake utilizado, a coluna 3 mostra o valor máximo assumido pela variável z e a coluna 4 mostra o procedimento de busca local adotado para esta versão.

Tabela 4.13: Versões implementadas com VNS

Notação	*Shake*	z_{max}	*Local Search*
VNS-5-5	1234P-BI-BW	5	1234P-FI-BW
VNS-2-5	1P-BI-BW	2	1234P-FI-BW
VNS-2-2	1P-BI-BW	2	1P-FI-BW

4.5 Instâncias utilizadas

Para validação do algoritmo proposto foram utilizadas as mesmas instâncias utilizadas em Calvi (2005) e Rizzato *et al.* (2010). O conjunto de instâncias é composto por duas das instâncias reais e oito instâncias aleatórias geradas por Calvi (2005) baseadas em uma instancia real.

Tabela 4.14: Instâncias utilizadas

Instância	Número de viagens	Tipo
AL130	130	Gerada aleatoriamente
AL251	251	Gerada aleatoriamente
RE412	412	Problema real
AL512	512	Gerada aleatoriamente
AL761	761	Gerada aleatoriamente
AL1000	1000	Gerada aleatoriamente
AL1253	1253	Gerada aleatoriamente
AL1512	1512	Gerada aleatoriamente
AL2010	2010	Gerada aleatoriamente
RE2313	2313	Problema real

As instâncias são descritas abaixo e representadas na Tabela 4.14, tendo como o prefixo (RE ou AL) para indicar se é uma instância real ou aleatória e a parte numérica indica o numero de viagens (*e.g.* RE412 é uma instância real e contém 412 tarefas):

- RE412 e RE2313: correspondem aos dados reais de duas empresas de transporte urbano de passageiros do Estado do Paraná;
- AL130, AL251, AL512, AL761, AL1000, AL1253, AL1517 e AL2010: são instâncias aleatórias obtidas a partir da instância RE2313 sorteando-se *blocos* da mesma.

Analisando a o tamanho das instâncias dos trabalhos relacionados da Tabela 2.2 da seção 2.2, observa-se que a maior instância foi utilizada por Silva e Cunha (2010) com 945 viagens. Do conjunto de instâncias utilizadas por este trabalho, metade delas contém um número de viagens superior a este valor, sendo que a maior instância possui 2,44 vezes o número de viagens utilizado por Silva e Cunha (2010).

5 Experimentos Computacionais

Este capítulo apresenta os resultados obtidos com os algoritmos propostos na seção 0 utilizando as instâncias descritas na seção 4.5.

Os algoritmos foram implementados em Pascal, no ambiente Lazarus V.1.0.12, e os experimentos realizados em dois ambientes. Os experimentos das implementações dos algoritmos determinísticos foram realizados em um microcomputador pessoal com processador *Intel Core* i5 M480 com clock de 2.67 GHz, 4GB de RAM e sistema operacional *Windows* 7. Os experimentos das implementações dos algoritmos com a meta-heurística VNS foram realizados no *Windows Server* 2008-R2, rodando em uma máquina virtual *KVM*, configurado para ocupar 30GB de RAM e 50 núcleos de um servidor com 4 processadores *Intel Xeon* E7-4860 (24MB de *cache* – 2.26 GHz) com sistema operacional *Linux CentOS* 6.

Os experimentos foram realizados inicialmente com as implementações dos algoritmos determinísticos utilizando todas as instâncias descritas na seção 4.5 e todas as versões descritas na seção 4.4.1.

Estes resultados foram utilizados para comparação com os resultados obtidos por Rizzato *et al.* (2012) e Calvi (2005) e também para determinar o melhor procedimento de busca local a ser utilizado na implementação com a meta-heurística VNS com as mesmas instâncias.

5.1 Resultados da abordagem determinística

Os melhores resultados obtidos utilizando esta abordagem são mostrados na Tabela 5.1. A coluna Jornadas (coluna 3) informa o número de motoristas necessários para realizar todas as viagens contidas na escala de veículos. A coluna Trocas (coluna 4) informa a quantidade de vezes que o motorista trocou de linha ou de veículo durante sua jornada de trabalho.

Tabela 5.1: Melhores resultados da abordagem determinística

Instância	Custo	Jornadas	Trocas	Versão	Iter.	Tempo
AL130	8.162,10	18	10	1P324-BI-BW	12	00:00:17
AL251	16.475,00	37	10	132P4-FI-BW	5	00:00:10
RE412	28.370,58	62	28	2413P-FI-BW	6	00:00:18
AL512	32.445,00	72	53	132P4-FI-FW	7	00:00:28
AL761	46.137,83	102	89	12P34-FI-BW	18	00:02:26
AL1000	62.553,25	140	68	123P4-FI-BW	6	00:01:44
AL1253	80.257,50	180	73	132P4-FI-BW	7	00:03:25
AL1517	97.895,00	220	73	12P34-FI-BW	8	00:06:11
AL2010	125.462,50	281	153	1324P-FI-BW	9	00:12:10
RE2313	141.343,65	315	198	1324P-FI-BW	14	00:25:32

Os valores relacionados na coluna Iterações (coluna 6) contabilizam o número de vezes que a sequência dos cinco procedimentos ilustrados na Figura 4.5 é executada até que a solução se estabilize.

Observamos que os melhores resultados para o custo da escala de jornadas não foram obtidos por uma única versão para todas as instâncias, mas podemos tirar algumas conclusões pela análise destes resultados, como:

- As versões que utilizam a técnica de busca local *First Improvement* e a sequência de cortes *Backward* apresentaram melhores resultados em todas as instâncias, com exceção da instância AL130.
- Para nenhuma instância o PCR aparece como o primeiro procedimento na fase de melhoramento.
- O 1-*swap* aparece como o primeiro procedimento para nove das dez instâncias utilizadas.
- É realizada uma média de 0,5 trocas por jornada dentro de uma escala, ou seja, para cada duas jornadas.

Uma análise complementar das melhores soluções é mostrada na Tabela 5.2. A porcentagem de minutos extras em relação aos minutos totais das jornadas varia entre 0,76% a 2,32%, com uma média de 1,36%. Um ponto importante está relacionado a não violação de restrições que apesar de não contribuírem para o aumento do custo total da escala, fazem com que a jornada de trabalho seja executada com maior satisfação pelo motorista, pois:

- O intervalo para descanso é respeitado entre 1 (uma) hora e 30 (trinta) minutos e 5 (cinco) horas;
- O tempo total de jornada nunca ultrapassa 13 (treze) horas, para que não ocorra interjornadas menor do que 11 (onze) horas;
- Uma jornada contínua de trabalho não ultrapassa 6 (seis) horas.

Tabela 5.2: Minutos extras e violações da abordagem determinística

Instância	Minutos Totais	Minutos Extras	% Min.extra	Violações		
				Intervalo	Máx. Jornada	Máx. Jor. Cont.
AL130	7.988,45	161,40	2,02%	0	0	0
AL251	15.876,75	130,00	0,82%	0	0	0
RE412	27.533,85	639,65	2,32%	0	0	0
AL512	31.615,35	510,00	1,61%	0	0	0
AL761	45.373,75	838,55	1,85%	0	0	0
AL1000	61.427,75	635,50	1,03%	0	0	0
AL1253	78.613,75	705,00	0,90%	0	0	0
AL1517	96.008,30	730,00	0,76%	0	0	0
AL2010	123.360,15	1215,00	0,98%	0	0	0
RE2313	139.344,45	1829,10	1,31%	0	0	0

Além dos melhores resultados para cada instância, também foram obtidos resultados comparativos, utilizando a ferramenta *Boxplot*, entre as diversas versões entre elas:

- Comparação entre as versões que utilizaram a técnica de busca *First Improvement* em relação ao *Best Improvement*;
- Comparação entre as versões que utilizaram a sequência de corte *Forward* em relação a sequência de corte *Backward*;
- Comparação entre as quatro combinações entre técnica de busca e sequências de corte utilizadas (FI-FW, FI-BW, BI-FW e BI-BW).

A Figura 5.1 mostra os resultados obtidos com as versões que utilizaram a técnica de busca *First Improvement* em relação às versões que utilizaram a técnica de busca *Best Improvement* para todas as instâncias.

A versão que utilizou a técnica *First Improvement* apresentou melhores resultados para todas as instâncias com exceção das instâncias AL130 e AL251.

A técnica *Best Improvement* apresentou resultados com menos esparsidade em relação ao *First Improvement*, principalmente nas duas maiores instâncias.

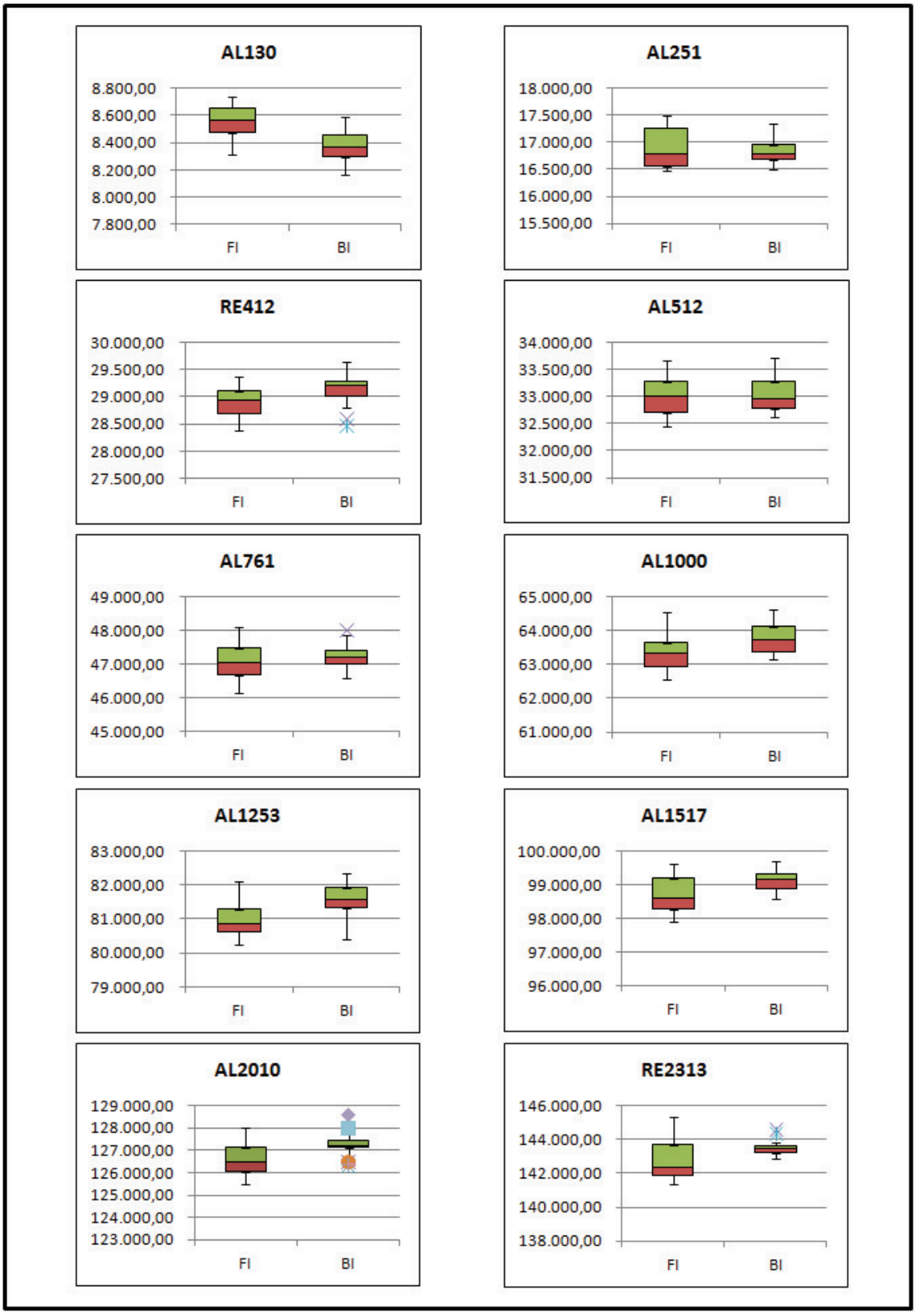

Figura 5.1: Comparativo entre as técnicas *First Improvement* x *Best Improvement* para todas as instâncias

A Figura 5.2 mostra os resultados obtidos com as versões que utilizaram a sequência de cortes *Forward* em relação às versões que utilizaram a sequência de cortes *Backward* para todas as instâncias.

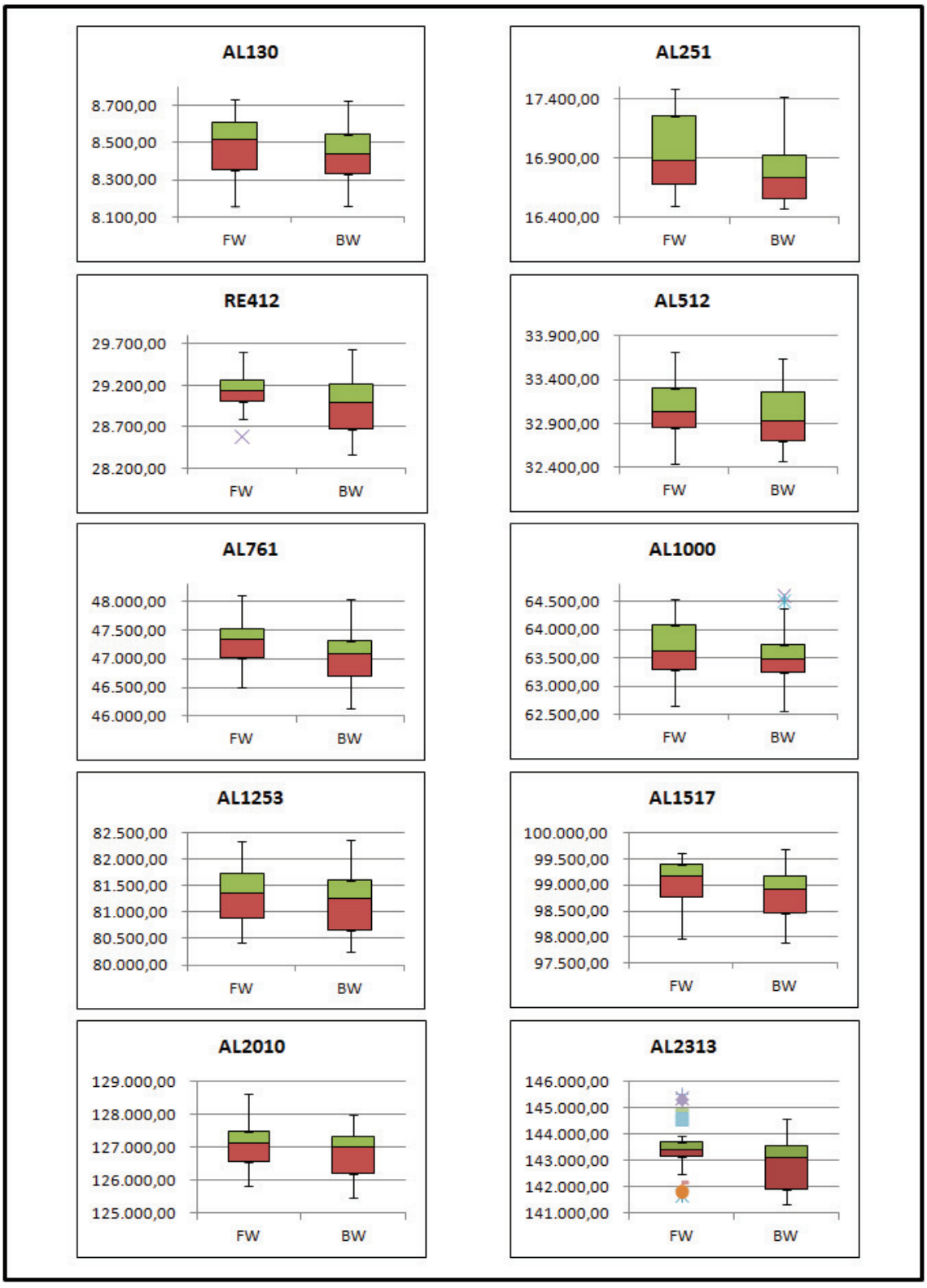

Figura 5.2: Comparativo entre as sequências *Forward* x *Backward* para todas as instâncias

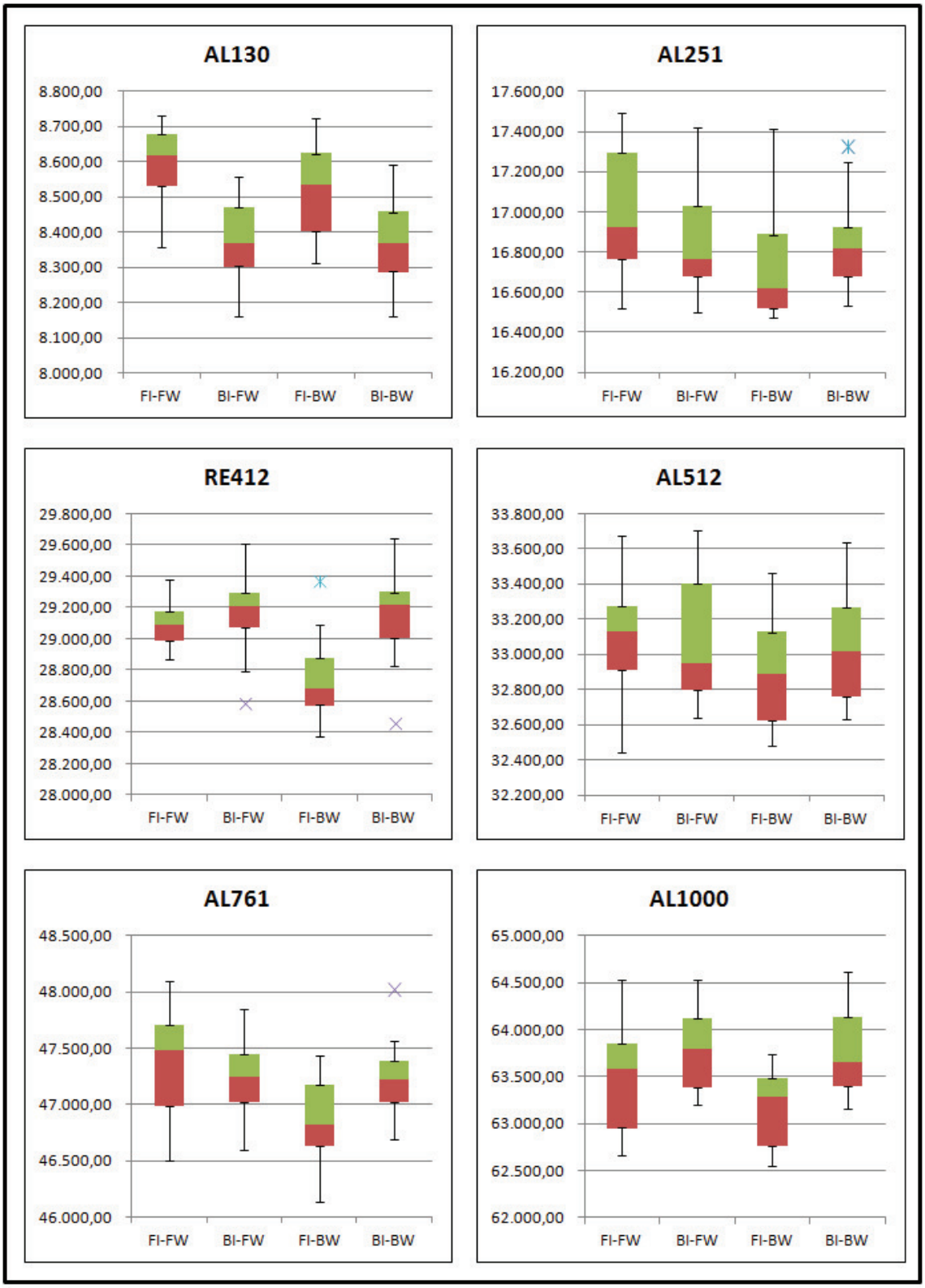

Figura 5.3: Resultados para abordagem determinística para instâncias AL130, AL251, RE412, AL512 e AL761

As implementações que utilizaram a sequência de cortes *Backward* apresentaram melhores resultados apesar de apresentarem maior esparsidade nas instâncias reais (RE412 e RE2313). Para a instância AL1517, a sequência *Backward* apresentou valor máximo acima do valor máximo da sequência *Forward*, contudo, a região interquartis apresentou melhores resultados.

A Figura 5.3 ilustra a comparação das versões combinando o método de busca e a sequência de cortes para as instâncias menores, isto é, AL130, AL251, RE412, AL512, AL761 e AL1000. Para as instâncias AL1253, AL1512, AL2010 e AL2030 a comparação é ilustrada pela Figura 5.4.

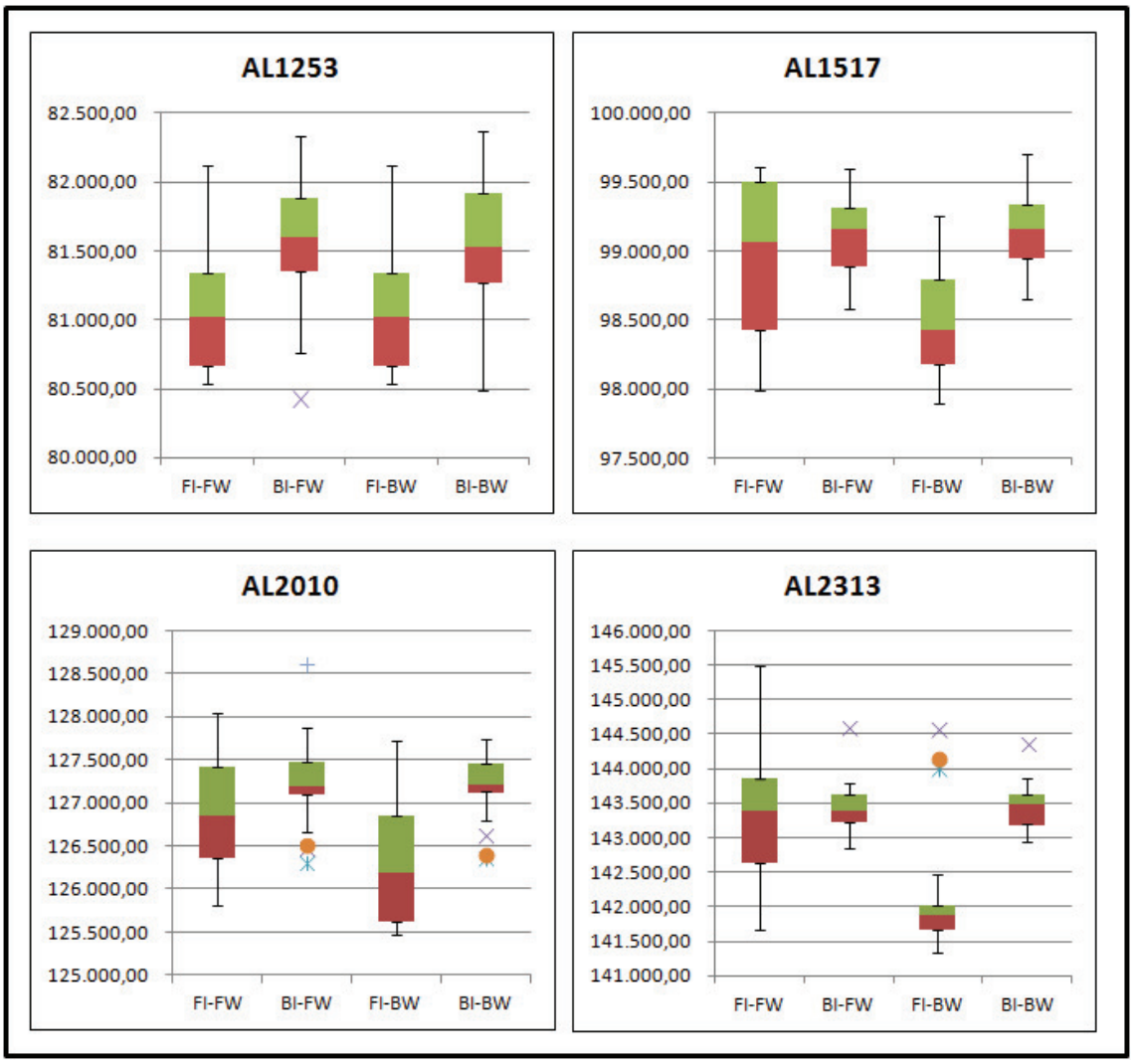

Figura 5.4: Resultados para abordagem determinística para instâncias AL1253, AL1517, AL2010 e RE2313

Para a instância AL130, as combinações utilizando o *Best Improvement* apresentaram melhores resultados, independente da sequência de cortes. A técnica *First Improvement*, independente da sequência de cortes, apresentou melhores resultados para a instância AL1253. Para as demais instâncias, os melhores resultados foram obtidos com a combinação *First Improvement-Backward*.

5.2 Comparação com resultados de Calvi e Rizzato

Nesta seção apresentamos um comparativo dos melhores resultados obtidos neste trabalho com os melhores resultados obtidos por outros trabalhos. Rizzato *el al.* (2012) utilizaram os procedimentos PCR e *k-swap* com a técnica *First Improvement* e sequência de cortes *Forward*. Calvi (2005) utilizou um procedimento denominado M1 que se assemelha ao procedimento PCR e M2 que realiza melhoramentos analisando as camadas em que possuam jornadas com horas extras.

Tabela 5.3: Comparação com resultados de Rizzato e Calvi

Instância	**Melhores resultados deste trabalho**		**Melhores resultados Rizzato (2012)**		**Melhores resultados Calvi (2005)**	
	N.Jor.	**Custo**	**N.Jor.**	**Custo**	**N.Jor.**	**Custo**
AL130	**18**	**8.162,10**	19	8.400,93	19	8.389,40
AL251	**37**	**16.475,00**	39	17.190,00	40	17.600,00
RE412	**62**	**28.370,58**	65	29.260,00	66	29.512,50
AL512	**72**	**32.445,00**	76	33.822,50	79	35.105,00
AL761	**102**	**46.137,83**	108	47.932,29	107	47.532,90
AL1000	**140**	**62.553,25**	144	64.097,29	146	64.873,60
AL1253	**180**	**80.257,50**	182	81.047,50	187	82.842,90
AL1517	**220**	**97.895,00**	221	98.283,21	225	99.852,80
AL2010	**281**	**125.462,50**	285	126.747,71	290	128.964,20
RE2313	**315**	**141.343,65**	327	145.701,43	331	147.215,00

A Tabela 5.3 ilustra que este os resultados obtidos por este trabalho apresentaram uma redução no número de jornadas da escala assim como uma redução no custo total das jornadas comparado aos resultados obtidos por Rizzato *el al.* (2012) e Calvi (2005) para todas as instâncias.

A Tabela 5.4 ilustra os *gaps* dos melhores resultados obtidos neste trabalho em relação aos melhores resultados obtidos por Rizzato *el al.* (2012) e Calvi (2005) quanto ao custo da escala de jornadas. *Gaps* negativos significam uma redução nos custos e *gaps* positivos significam um aumento nos custos. O maior *gap* encontrado em relação aos trabalhos citados foi obtido com a instância AL512 e o menor com a instância AL1517.

Tabela 5.4: *Gap* para os resultados de Rizzato e Calvi

Instância	Melhores resultados deste trabalho	Melhores resultados Rizzato (2012)	*Gap* Rizzato	Melhores resultados Calvi (2005)	*Gap* Calvi
AL130	**8.162,10**	8.400,93	-2,84%	8.389,40	-2,71%
AL251	**16.475,00**	17.190,00	**-4,16%**	17.600,00	-6,39%
RE412	**28.370,58**	29.260,00	-3,04%	29.512,50	-3,87%
AL512	**32.445,00**	33.822,50	-4,07%	35.105,00	**-7,58%**
AL761	**46.137,83**	47.932,29	-3,74%	47.532,90	-2,93%
AL1000	**62.553,25**	64.097,29	-2,41%	64.873,60	-3,58%
AL1253	**80.257,50**	81.047,50	-0,97%	82.842,90	-3,12%
AL1517	**97.895,00**	98.283,21	-0,39%	99.852,80	-1,96%
AL2010	**125.462,50**	126.747,71	-1,01%	128.964,20	-2,72%
RE2313	**141.343,65**	145.701,43	-2,99%	147.215,00	-3,99%

5.3 Resultados da abordagem com VNS

Para obtenção de resultados significativos desta abordagem, foram realizados experimentos que consistiram em rodar um total de 50 (cinquenta) iterações de cada uma das três versões implementadas.

A Figura 5.5 ilustra os resultados obtidos com este experimento através de gráficos boxplot para as instâncias AL130, AL251, RE412, AL512, AL761 e AL1000. A Figura 5.6 ilustra os mesmos gráficos para as instâncias AL1253, AL1517, AL2010 e RE2313.

Observa-se que a versão VNS-2-2 apresentou piores resultados em relação às versões VNS-5-5 e VNS-2-5 para todas as instâncias. A versão VNS-5-5 apresentou melhores resultados em relação à versão VNS-2-5, exceto na instância AL130.

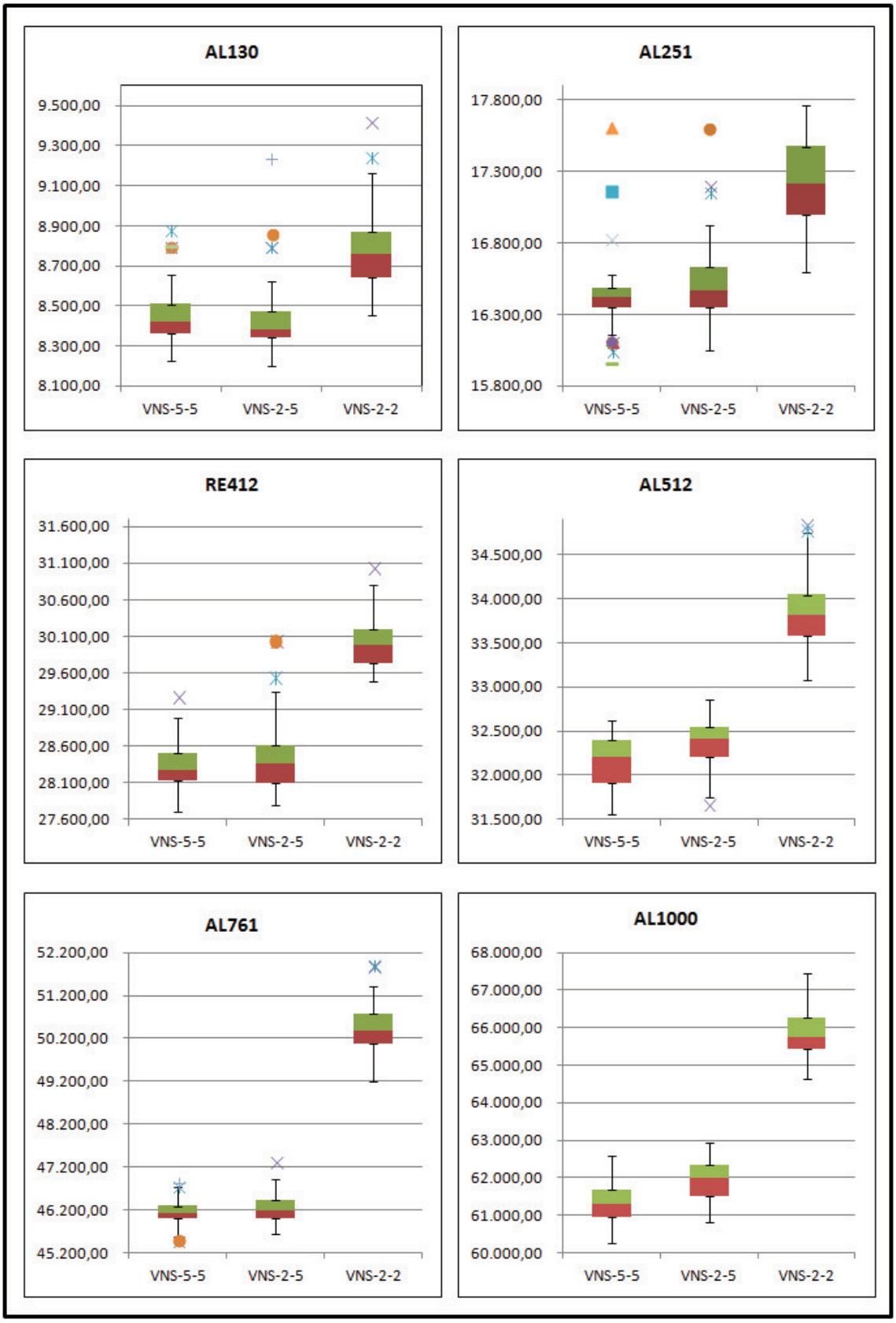

Figura 5.5: Resultados da abordagem com VNS para as instâncias AL130, AL251, RE412, AL512, AL761 e AL1000

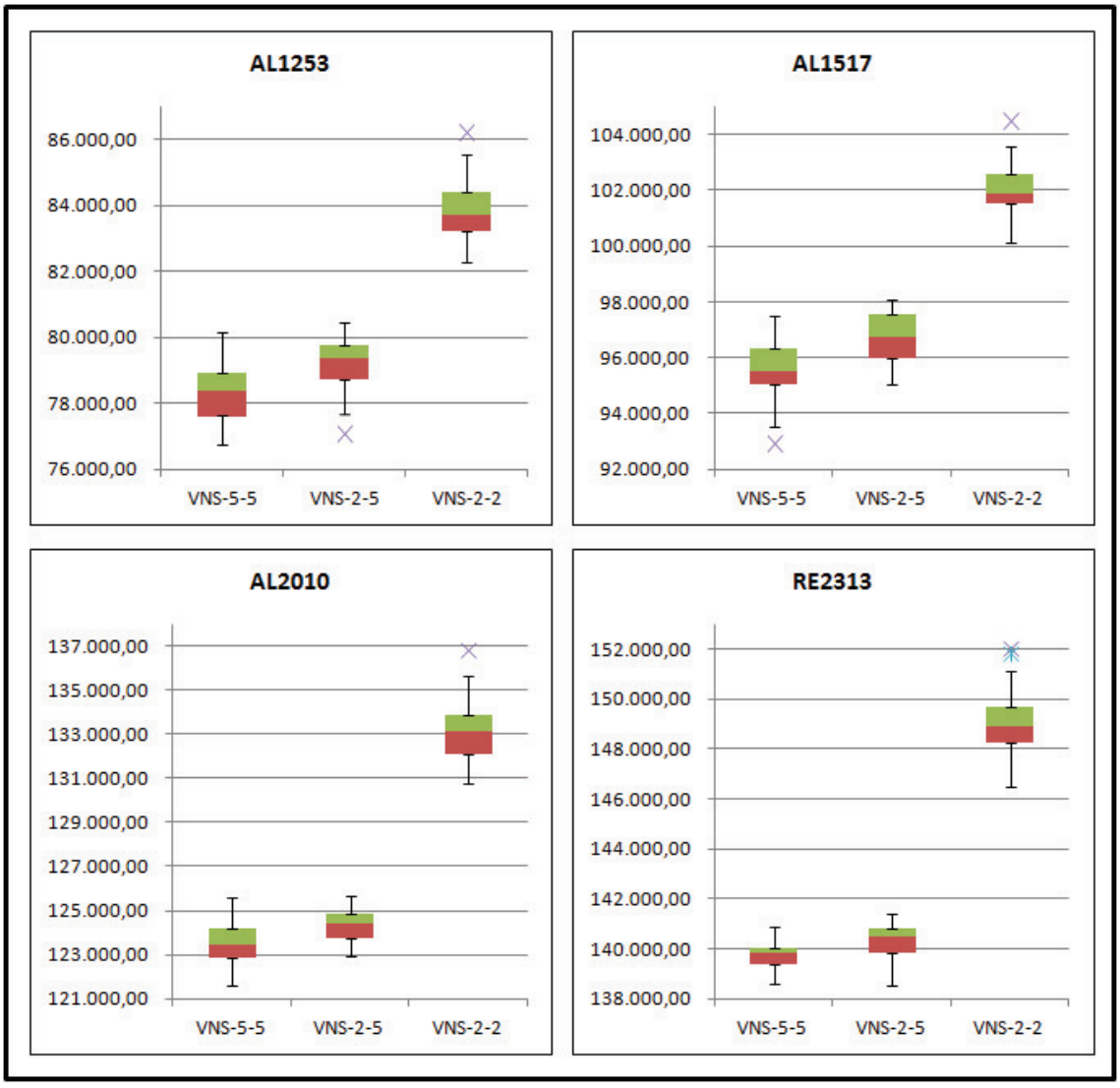

Figura 5.6: Resultados da abordagem com VNS para as instâncias AL1253, AL1517, AL2010 e RE2313

A Tabela 5.5 mostra os valores da mediana e o número médio de jornadas de cada uma das instâncias para os experimentos realizados com as três versões, dando uma visão numérica dessas diferenças.

A Tabela 5.6 mostra o tempo médio de cada iteração para as três versões que utilizam a meta-heurística VNS. O cálculo do tempo médio foi feito utilizando-se o tempo total da simulação dividido pelo número de iterações.

Tabela 5.5: Custos médios e número médio de jornadas para a abordagem com VNS

Instância	VNS-5-5	Jorn	VNS-2-5	Jorn	VNS-2-2	Jorn
AL130	8.420,00	19	**8.381,59**	**18**	8.756,45	19
AL251	**16.421,23**	**37**	16.469,26	**37**	17.219,58	38
RE412	**28.278,71**	63	28.363,75	**62**	29.980,00	65
AL512	**32.200,08**	**71**	32.417,25	72	33.817,20	72
AL761	**46.120,61**	**102**	46.172,50	**102**	50.373,31	106
AL1000	**61.293,70**	**137**	61.998,73	139	65.743,73	143
AL1253	**78.385,71**	**174**	79.388,73	178	83.726,94	181
AL1517	**95.525,01**	**213**	96.713,45	216	101.905,68	221
AL2010	**123.443,45**	**275**	124.418,50	278	133.119,29	283
RE2313	**139.809,35**	**310**	140.532,20	313	148.879,23	317

Tabela 5.6: Tempo médio de uma iteração para cada uma das versões com VNS

Instância	VNS-5-5	VNS-2-5	VNS-2-2
AL130	00:07:57	00:04:20	00:01:58
AL251	00:14:18	00:06:20	00:03:11
RE412	00:19:50	00:09:36	00:04:59
AL512	00:25:14	00:10:54	00:05:26
AL761	00:44:41	00:23:07	00:13:41
AL1000	01:29:59	00:43:37	00:24:05
AL1253	03:47:51	01:24:07	00:40:47
AL1517	05:43:21	02:23:01	01:15:59
AL2010	12:04:57	04:41:16	02:26:59
RE2313	15:43:24	06:20:58	02:57:19

Confirmando as informações da Figura 5.5, observamos que a versão VNS-2-2 apresentou valores bem superiores em relação às versões VNS-5-5 e VNS-2-5, apesar do tempo de execução ser inferior. As versões VNS-5-5 e VNS-2-5 apresentaram custos da escala próximos e uma grande diferença no tempo médio de execução.

A

Tabela **5.7** mostra os *gaps* de custo da escala e a Tabela 5.8 mostra os *gaps* de tempo de execução médio da versão VNS2-5 em relação à versão VNS-5-5.

As tabelas mostram que a versão VNS-2-5 apresentou custos superiores entre 0,29% à 1,28%, com exceção da instância AL130, em troca de um tempo de execução que na média corresponde a 55,18% do tempo de execução da versão VNS-5-5.

Tabela 5.7: Comparação de custos entre as versões VNS-5-5 e VNS-2-5

Instância	Custo VNS-5-5	Custo VNS-2-5	*Gap* Custo
AL130	8.420,00	**8.381,59**	0,46%
AL251	**16.421,23**	16.469,26	-0,29%
RE412	**28.278,71**	28.363,75	-0,30%
AL512	**32.200,08**	32.417,25	-0,67%
AL761	**46.120,61**	46.172,50	-0,11%
AL1000	**61.293,70**	61.998,73	-1,15%
AL1253	**78.385,71**	79.388,73	-1,28%
AL1517	**95.525,01**	96.713,45	-1,24%
AL2010	**123.443,45**	124.418,50	-0,79%
RE2313	**139.809,35**	140.532,20	-0,52%

Tabela 5.8: Comparação de tempos de execução médio entre as versões VNS-5-5 e VNS-2-5

Instância	Tempo VNS-5-5	Tempo VNS-2-5	*Gap* Tempo
AL130	00:07:57	00:04:20	45,54%
AL251	00:14:18	00:06:20	55,75%
RE412	00:19:50	00:09:36	51,60%
AL512	00:25:14	00:10:54	56,83%
AL761	00:44:41	00:23:07	48,28%
AL1000	01:29:59	00:43:37	51,53%
AL1253	03:47:51	01:24:07	63,08%
AL1517	05:43:21	02:23:01	58,35%
AL2010	12:04:57	04:41:16	61,20%
RE2313	15:43:24	06:20:58	59,62%

Como os tempos de execução das versões que utilizam a meta-heurística VNS foram relativamente altos, foi realizado experimento para obter a convergência da solução obtida para conhecermos o momento em que a solução final foi obtida e o processo ainda continua em execução sem melhoras.

Para a instância RE2313, foram levantados os gráficos de convergência para as versões VNS-5-5 e VNS-2-5 com um experimento realizado. A Figura 5.7 ilustra a convergência da solução para VNS-5-5 e a Figura 5.8 ilustra a convergência da solução para a versão VNS-2-5.

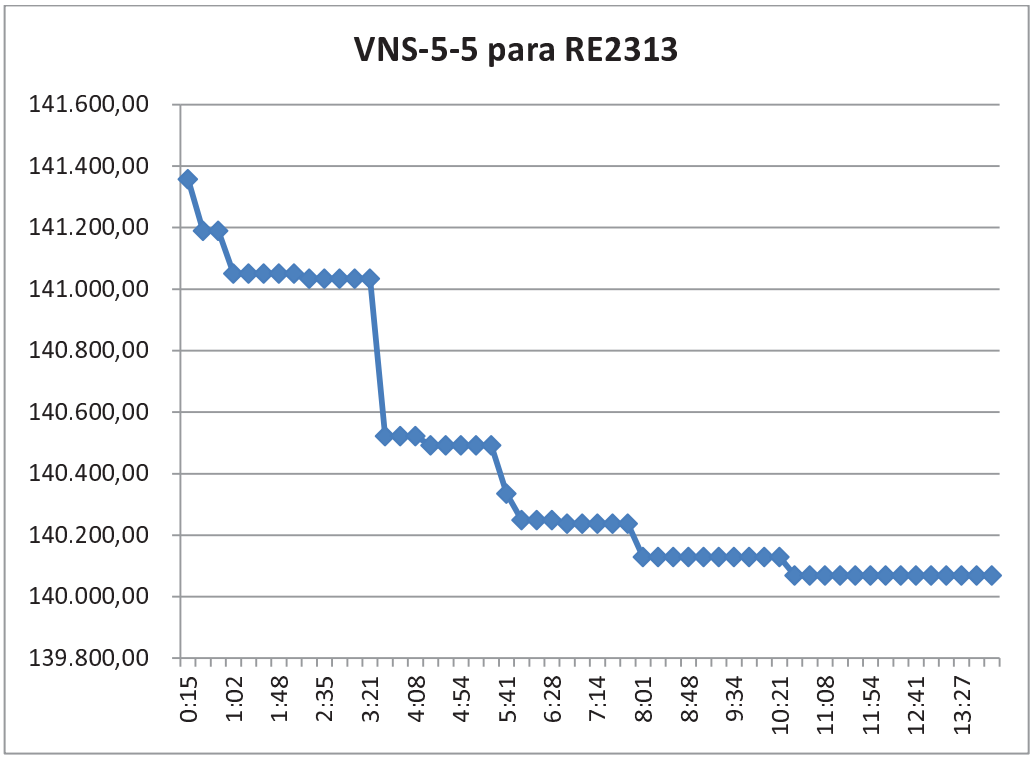

Figura 5.7: Gráfico de Convergência para VNS-5-5

O primeiro valor obtido no gráfico da Figura 5.7 corresponde ao valor da solução obtida após a primeira iteração do processo. Como o procedimento *Shake* utiliza os procedimentos 1-*swap*, 2-*swap*, 3-*swap*, 4-*swap* e PCR em sua construção, devemos ter no máximo cinco iterações sem que haja melhora na solução. Nos últimos dois degraus do gráfico, observamos uma estabilidade na solução para mais de cinco iterações, com dez iterações para o penúltimo degrau e quatorze iterações para o último degrau. Nestes dois casos, apesar do custo da escala ser o mesmo, está havendo uma melhora na qualidade da solução, com menos trocas de veículos ou linhas pelos motoristas.

Se for levado em conta somente o custo da escala, sem se importar com a quantidade de trocas de veículos ou de linhas pelos motoristas, o processo obteve o melhor resultado aproximadamente a três horas antes do tempo total da simulação.

Para a versão VNS-2-5, ilustrado na Figura 5.8, temos somente duas iterações sem que haja melhora na solução, pois o procedimento *Shake* utiliza somente o 1-*swap* e o PCR. Somente para o último degrau, observamos a ocorrência de seis iterações para a solução final, com a mesma justificativa apresentada para a versão VNS-5-5. Considerando somente o valor do custo da escala, a melhor solução foi obtida aproximadamente duas horas antes do tempo total da simulação.

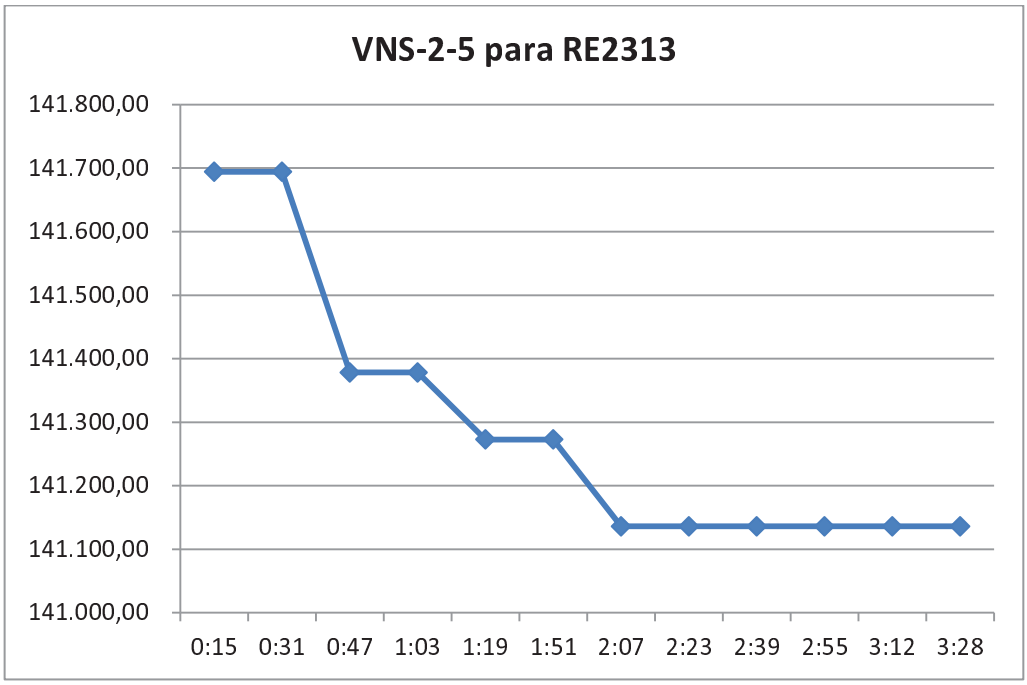

Figura 5.8: Gráfico de Convergência para VNS-2-5

Os melhores resultados obtidos para cada uma das versões estão ilustrados na Tabela 5.9. Estes valores estão levando em conta os pontos *outliers* do gráfico da Figura 5.5. A versão VNS-5-5 apresentou melhores custos em todas as instâncias, exceto para as instâncias AL130 e RE2313.

Tabela 5.9: Melhores resultados obtidos com VNS

Instância	VNS-5-5	Jorn.	VNS-2-5	Jorn.	VNS-2-2	Jorn.
AL130	8.224,95	18	**8.196,08**	18	8.451,23	18
AL251	**15.959,73**	**35**	16.048,58	36	16.593,60	36
RE412	**27.695,15**	**61**	27.792,50	**61**	29.490,00	64
AL512	**31.552,80**	**69**	31.665,90	**69**	33.084,05	70
AL761	**45.467,53**	**100**	45.652,55	**100**	49.199,95	107
AL1000	**60.256,58**	**132**	60.811,88	135	64.627,35	141
AL1253	**76.739,53**	**169**	77.123,48	178	82.258,70	175
AL1517	**92.952,50**	**205**	95.039,35	212	100.109,60	217
AL2010	**121.578,95**	**268**	122.972,35	275	130.729,90	278
RE2313	138.608,03	**303**	**138.507,98**	306	146.461,10	316

A Tabela 5.10 mostra detalhes das melhores escalas obtidas com a abordagem utilizando VNS. Em todas as escalas geradas, nenhuma das jornadas apresentou violação de:

- Intervalo menor que 1 (uma) hora e 30 (trinta) minutos e maior que 5 (cinco) horas;
- Jornada de trabalho máximo menor que 13 (treze) horas;
- Jornada de trabalho contínua menor que 6 (seis) horas .

Tabela 5.10: Detalhes dos melhores resultados obtidos com VNS

Instância	Custo	Jornadas	Trocas	Minutos Totais	Minutos Extras	% Min.extra
AL130	8.196,08	18	14	8038	184	2,29%
AL251	15.959,73	35	32	15642	373	2,39%
RE412	27.695,15	61	35	26935	570	2,12%
AL512	31.552,80	69	59	30957	795	2,57%
AL761	45.467,53	100	106	44784	978	2,18%
AL1000	60.256,58	132	112	51485	1451	2,82%
AL1253	76.739,53	169	155	75449	1586	2,10%
AL1517	92.952,50	205	171	91491	1835	2,01%
AL2010	121.578,95	268	225	121579	2439	2,01%
RE2313	138.507,98	306	325	136760	2579	1,89%

5.4 Comparação entre abordagem determinística e abordagem com VNS

Os melhores resultados obtidos com a abordagem utilizando VNS não podem ser utilizados para serem comparados com os melhores resultados obtidos com a abordagem determinística. Isto se deve ao fato da abordagem que utiliza a meta-heurística VNS ser uma abordagem que utiliza recursos randômicos para geração de estruturas de vizinhança que geralmente não são repetitivos.

Para esta comparação é utilizado o boxplot com a união dos gráficos obtidos na Figura 5.3 e Figura 5.4 que mostram os resultados obtidos para as quatro versões da abordagem determinística e da Figura 5.5 que mostra os resultados obtidos para as três versões da abordagem utilizando a meta-heurística VNS.

A Figura 5.9 ilustra os resultados para a instância AL130. Os resultados obtidos pela abordagem determinística foram melhores que os resultados obtidos pela

abordagem que utiliza a meta-heurística VNS. As versões que utilizaram a técnica *Best Improvement* produziram menores custos em relação às versões que utilizaram a técnica *First Improvement*. Das versões que utilizaram o VNS, a versão VNS2-5 apresentou o melhor resultado e pior resultado foi obtido com a versão VNS-2-2.

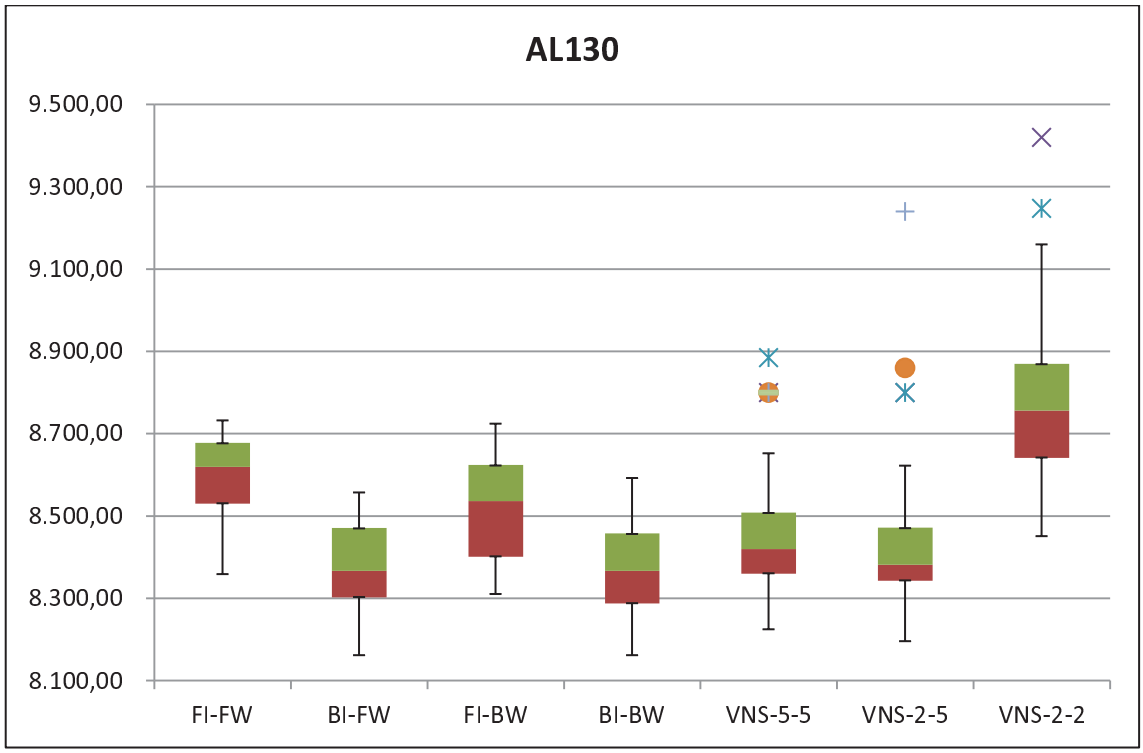

Figura 5.9: Comparação da abordagem determinística com a abordagem que utiliza VNS para a instância AL130

A Figura 5.10 ilustra os resultados obtidos para a instância AL251. Os melhores resultados foram obtidos com a versão VNS-5-5, apesar de apresentar alguns pontos *outliers*. A versão VNS-2-2 apresentou os piores resultados sendo superada por todas as versões da abordagem determinística.

Para as instâncias RE412, AL512, AL761, AL1000, AL1253, AL1517, AL2010 e RE2313 os melhores resultados foram obtidos com a versão VNS-5-5 seguido da versão VNS-2-5 e os piores resultados com a versão VNS-2-2, conforme ilustram as figuras a seguir (de 5.11 a 5.18, respectivamente).

Comparando somente as versões determinísticas, a versão FI-BW obteve os melhores resultados para as instâncias AL251, RE412, AL761, AL1000, AL1517, AL2010 e RE2313. Para a instância AL512 a versão FI-FW obteve o melhor resultado e a versão BI-BW obteve o melhor resultado para a instância AL1253.

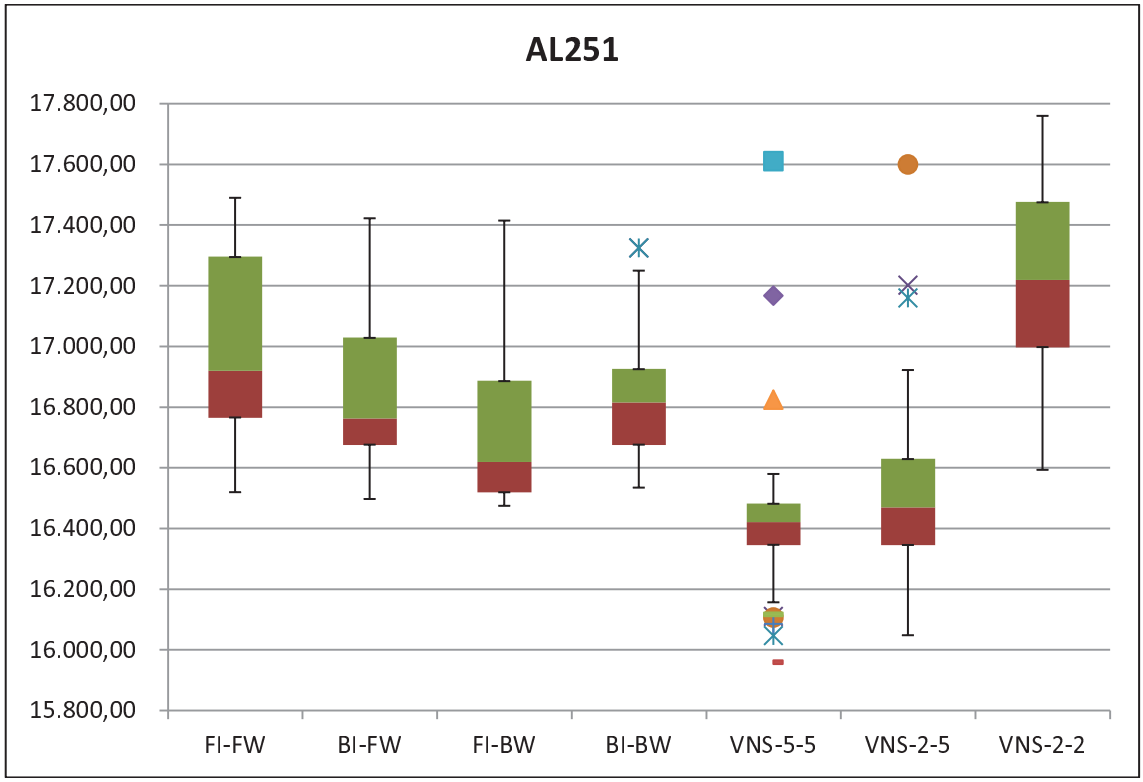

Figura 5.10: Comparação da abordagem determinística com a abordagem que utiliza VNS para a instância AL251

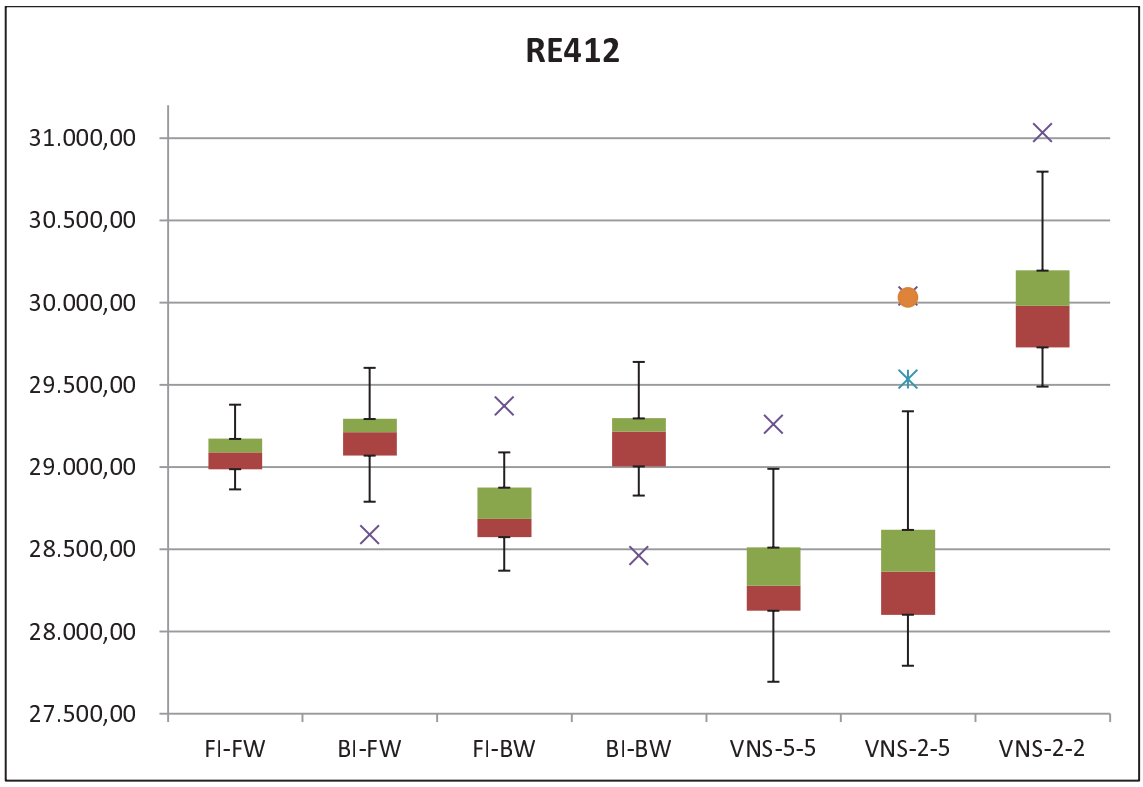

Figura 5.11: Comparação da abordagem determinística com a abordagem que utiliza

VNS para a instância RE412

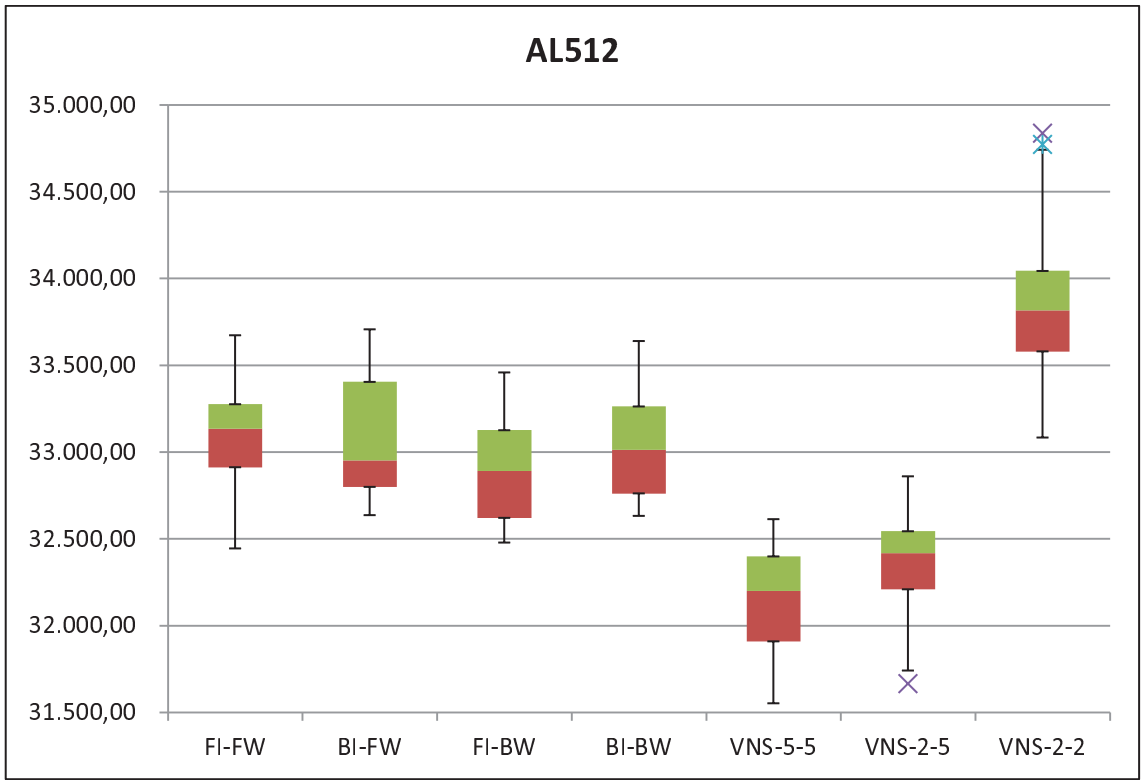

Figura 5.12: Comparação da abordagem determinística com a abordagem que utiliza VNS para a instância AL512

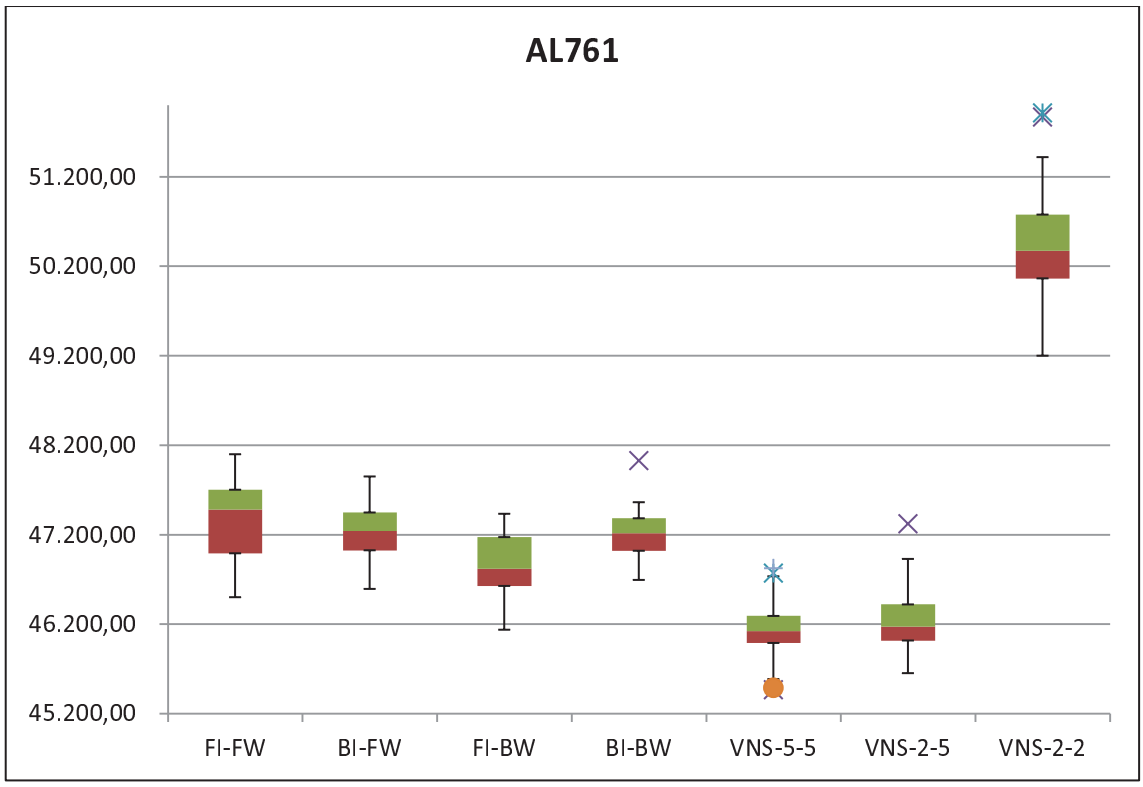

Figura 5.13: Comparação da abordagem determinística com a abordagem que utiliza

VNS para a instância AL761

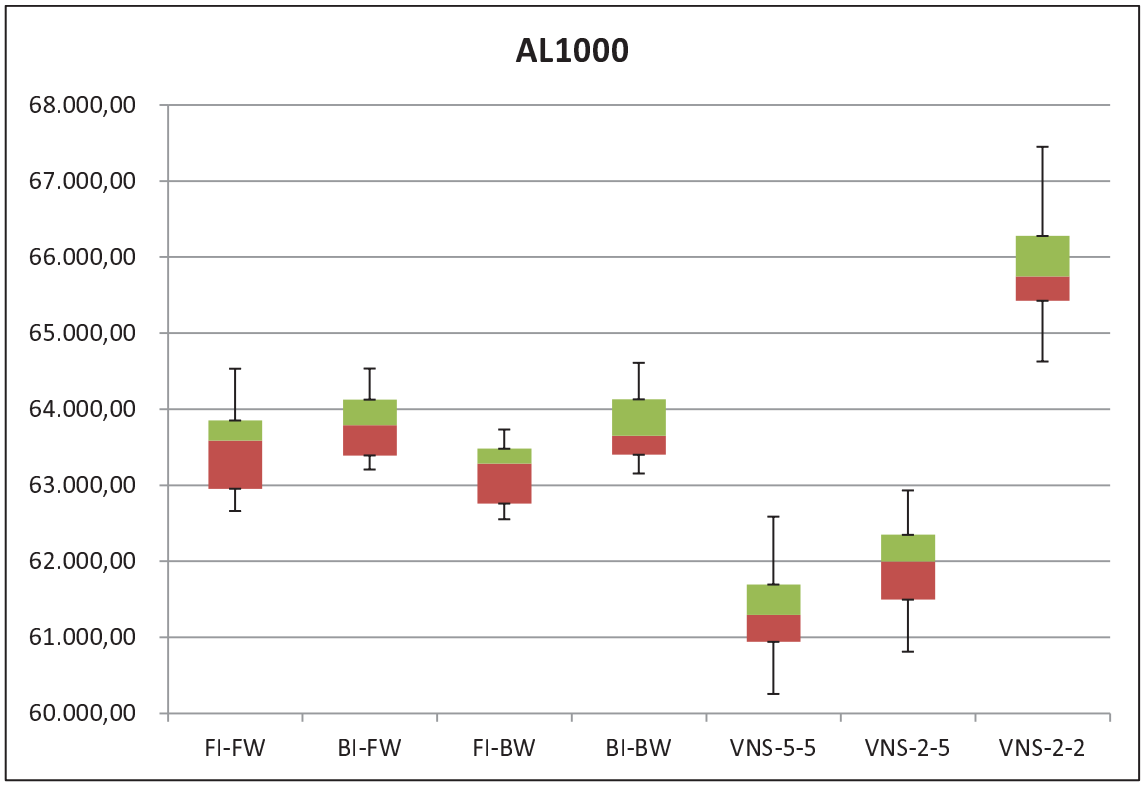

Figura 5.14: Comparação da abordagem determinística com a abordagem que utiliza VNS para a instância AL1000

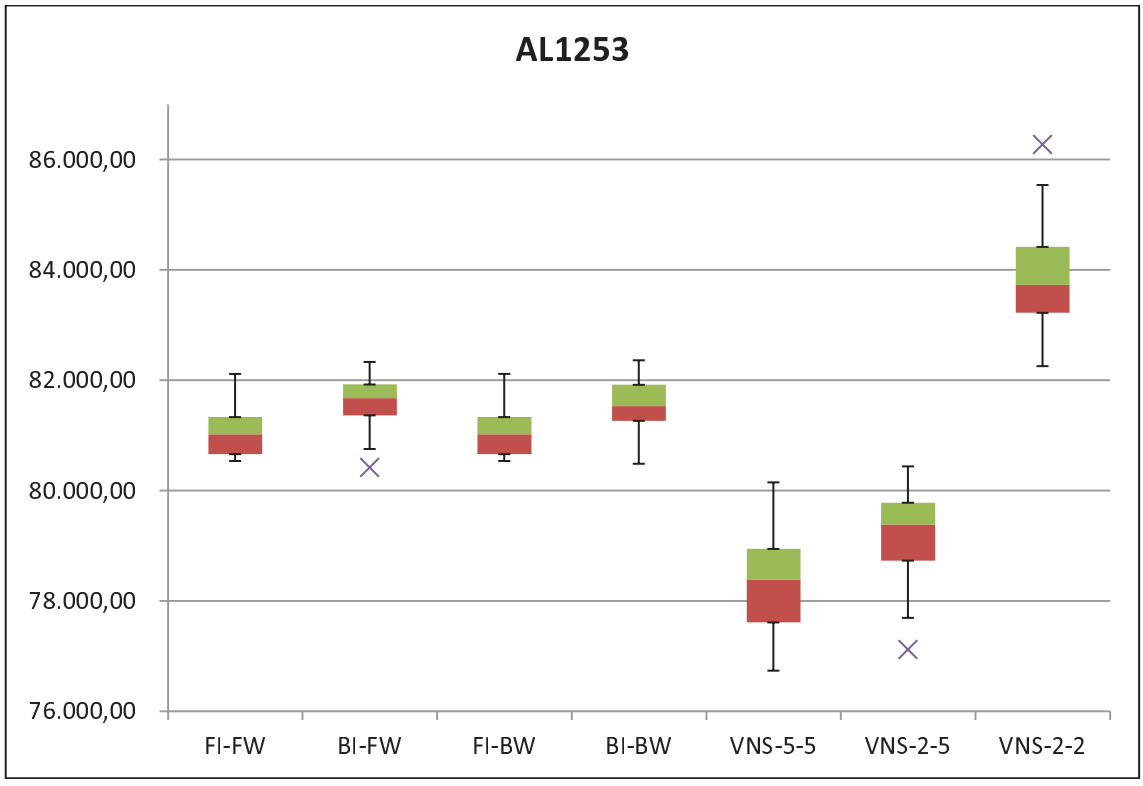

Figura 5.15: Comparação da abordagem determinística com a abordagem que utiliza

VNS para a instância AL1253

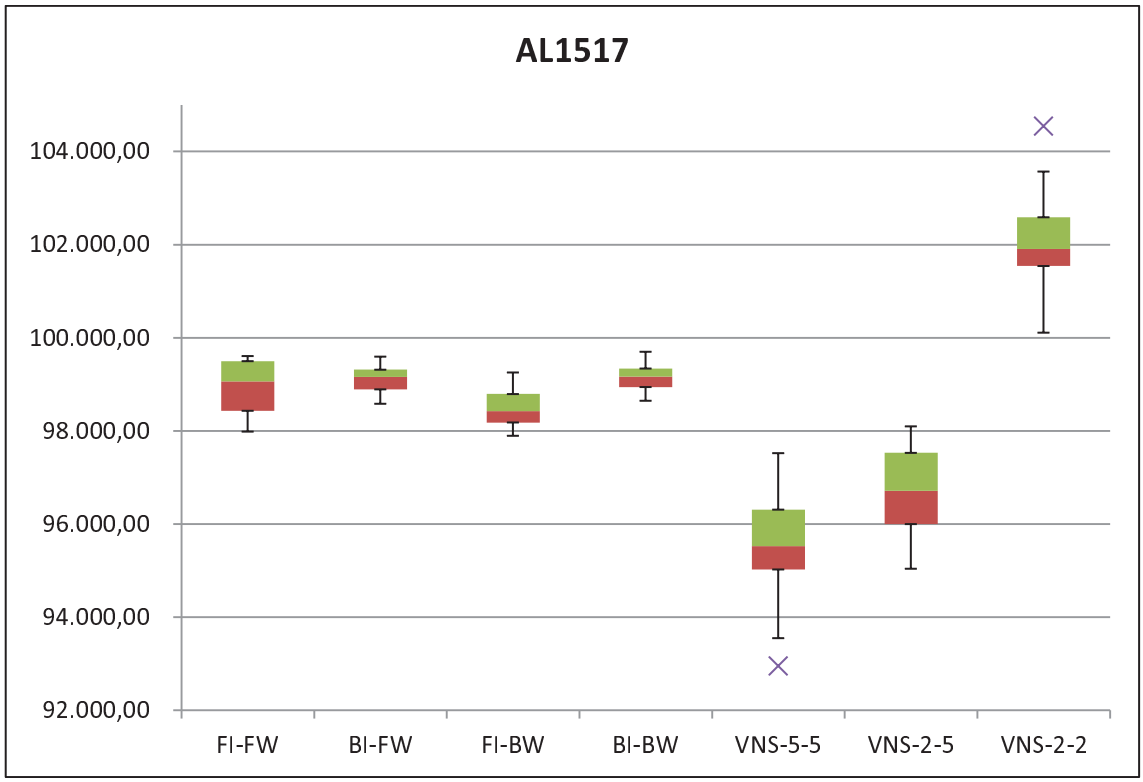

Figura 5.16: Comparação da abordagem determinística com a abordagem que utiliza VNS para a instância AL1517

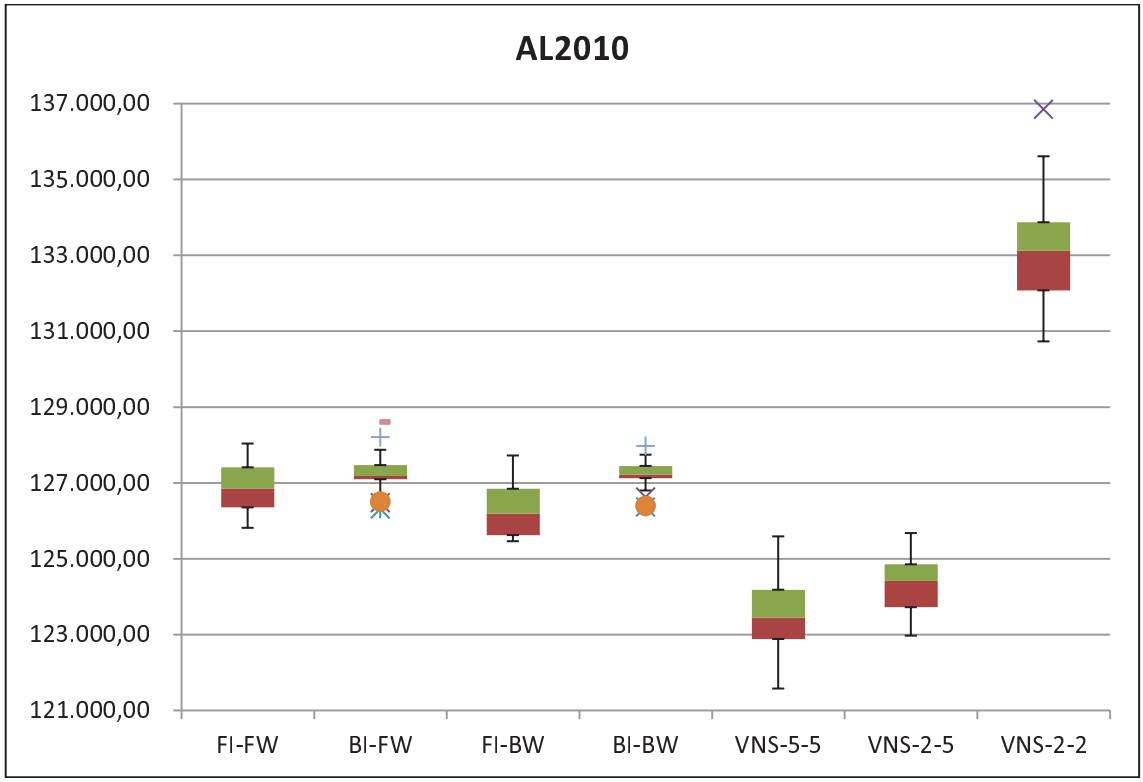

Figura 5.17: Comparação da abordagem determinística com a abordagem que utiliza

VNS para a instância AL2010

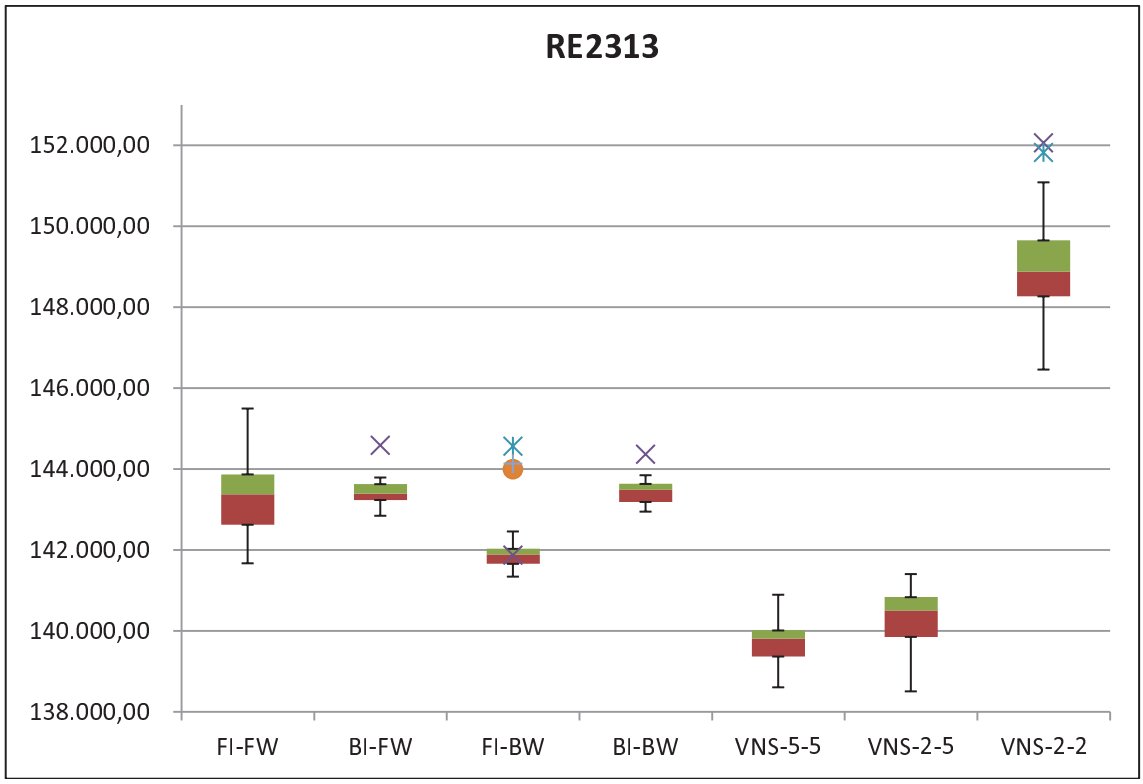

Figura 5.18: Comparação da abordagem determinística com a abordagem que utiliza VNS para a instância RE2313

Os experimentos com a abordagem utilizando a meta-heurística VNS apresentaram melhores resultados para todas as instâncias, com exceção da instância AL130.

A versão VNS-2-2 apresentou os piores resultados para todas as instâncias, apresentando diferenças maiores para instâncias maiores, inclusive em relação aos resultados obtidos pela abordagem determinística.

A Tabela 5.11 ilustra uma comparação numérica obtida entre a abordagem determinística e as versões VNS-5-5 e VNS-2-5, relativa aos custos da escala e número de jornadas. Para o custo da escala das versões com VNS, foram inseridos os valores médios, ou seja, o valor da mediana. A Tabela 5.12 apresenta os valores dos *gaps* obtidos comparando os custos e o número de jornadas obtidas com as versões VNS-5-5 e VNS-2-5 em relação a abordagem determinística. Valores positivos de *gap* significa aumento de custo ou número de jornadas e valores negativos de *gap* significa redução de custo ou número de jornadas.

Tabela 5.11: Melhores resultados entre abordagem determinística, VNS-5-5 e VNS-2-5

Instância	Abord. Determ.		VNS-5-5		VNS-2-5	
	Custo	Jorn.	Custo	Jorn.	Custo	Jorn.
AL130	**8.162,10**	**18**	8.420,00	19	8.381,59	18
AL251	16.475,00	37	**16.421,23**	37	16.469,26	37
RE412	28.370,58	**62**	**28.278,71**	63	28.363,75	**62**
AL512	32.445,00	72	**32.200,08**	**71**	32.417,25	72
AL761	46.137,83	102	**46.120,61**	102	46.172,50	102
AL1000	62.553,25	140	**61.293,70**	**137**	61.998,73	139
AL1253	80.257,50	180	**78.385,71**	**174**	79.388,73	178
AL1517	97.895,00	220	**95.525,01**	**213**	96.713,45	216
AL2010	125.462,50	281	**123.443,45**	**275**	124.418,50	278
RE2313	141.343,65	315	**139.809,35**	**310**	140.532,20	313

Tabela 5.12: *Gap* do custo e do número de jornadas entre as versões VNS-5-5 e VNS-2-5 e a abordagem determinística

Instância	VNS-5-5 x Determ.		VNS-2-5 x Determ.	
	Gap Custos	*Gap* Jorn.	*Gap* Custos	*Gap* Jorn.
AL130	3,16%	5,56%	2,69%	0,00%
AL251	-0,33%	0,00%	-0,03%	0,00%
RE412	-0,32%	1,61%	-0,02%	0,00%
AL512	-0,75%	-1,39%	-0,09%	0,00%
AL761	-0,04%	0,00%	0,08%	0,00%
AL1000	-2,01%	-2,14%	-0,89%	-0,71%
AL1253	-2,33%	**-3,33%**	-1,08%	-1,39%
AL1517	**-2,42%**	-3,18%	**-1,21%**	**-1,82%**
AL2010	-1,61%	-2,14%	-0,83%	-1,07%
RE2313	-1,09%	-1,59%	-0,57%	-0,63%

Comparando os melhores resultados obtidos com a abordagem determinística em relação aos resultados médios obtidos com a versão VNS-5-5, temos:

- Os maiores *gaps* obtidos em relação ao custo foram com as instâncias AL1517, AL1253 e AL1000. Em relação ao número de jornadas, os maiores *gaps* foram para as mesmas três instâncias mais a instância AL2010;
- Para a instância AL130, a abordagem determinística apresentou resultados melhores tanto quanto a custo como também em relação ao número de jornadas;

- Houve um decréscimo do *gap* para as instâncias AL2010 e RE2313 em relação à instância AL1517.

Comparando os melhores resultados obtidos com a abordagem determinística em relação aos resultados médios obtidos com a versão VNS-2-5, temos:

- Os maiores *gaps* obtidos em relação ao custo foram com as instâncias AL1517, AL1253 e AL1000. Em relação ao número de jornadas, os maiores *gaps* foram para instâncias AL1517, AL1253 e AL2010;
- A abordagem determinística obteve melhores resultados de custo para as instâncias AL130 e Al761;
- Houve uma diminuição no *gap* para as instâncias AL2010 e RE2313 em relação à instância AL1517.

A versão VNS-5-5 apresentou *gaps* maiores tanto no custo da escala como também em relação ao número de jornadas em relação à versão VNS-2-5 quando comparados com os melhores resultados obtidos pela abordagem determinística.

Observa-se um pico de *gap* para os custos ocorrendo na instância AL1517 para as duas comparações. Já em relação ao número de jornadas o pico no *gap* ocorreu na instância AL1253 quando comparando os resultados da versão VNS-5-5 com os melhores resultados da abordagem determinística. Na comparação da versão VNS-2-5 com os melhores resultados da versão determinística o *gap* máximo em relação ao número de jornadas ocorreu na instância AL1517.

6 Conclusão

O Problema de Escalonamento de Motoristas (PEM) é um problema de grande importância no planejamento operacional de empresas do setor de transporte público, pois os custos relacionados com recursos humanos, a maioria motoristas, contribuem com uma grande parcela do custo total.

Mauri (2005) e Marinho (2005) relatam que muitas empresas de transporte público ou privado utilizam métodos manuais para geração da escala de motoristas, demandando muito tempo e nem sempre obtendo boas soluções.

Algoritmos exatos foram desenvolvidos para resolução deste problema, mas esta técnica possui limitações para atender a instâncias de grande porte, conforme ilustra a Tabela 2.1 da seção 2.2. Algoritmos heurísticos foram desenvolvidos para solucionar o PEM para instâncias de grande porte, conforme ilustra a Tabela 2.2 da seção 2.2.

Este trabalho teve como objetivo propor uma nova abordagem utilizando os mesmos procedimentos de busca local PCR e *k-swap* utilizados nos trabalhos propostos por Calvi (2005) e Rizzato *et al.* (2012).

Os procedimentos PCR e *k-swap* foram utilizados com técnicas de busca local *First Improvement* e *Best Improvement*, com sequências de corte *Forward* e *Backward* e também em conjunto com a meta-heurística VNS.

Para solucionar o PEM foram implementadas e avaliadas sete versões de algoritmos utilizando os procedimentos PCR e *k*-swap em duas abordagens. Quatro versões são baseadas em uma abordagem determinística, descritas na seção 4.4.1, e três versões em uma abordagem utilizando a meta-heurística VNS descritas na seção 4.4.2. A escolha pela utilização da meta-heurística VNS foi baseada na característica desta meta-heurística de trabalhar em conjunto com outros procedimentos de busca local em sua estrutura, como o PCR e *k-swap*.

Na abordagem determinística, as versões que utilizam a técnica de busca local *First Improvement* e a sequência de cortes *Backward* apresentaram melhores resultados em todas as instâncias, com exceção da instância AL130. Os resultados obtidos por esta

abordagem apresentaram melhores resultados comparados aos resultados obtidos por Calvi (2005) e Rizzato *et al.* (2012), para todas as instâncias. Em relação a Rizzato (2012), foi obtido uma redução entre 0,39% e 4,16% nos custos da escala de jornadas. Já em relação a Calvi (2005), a redução nos custos ficou entre 1,96% e 7,58%.

Este trabalho contribuiu dando continuidade no desenvolvimento de algoritmos que utilizam os procedimentos PCR e *k-swap*, ampliando as opções de utilização destes procedimentos dentro da abordagem determinística e em conjunto com a meta-heurística VNS, obtendo melhores resultados em relação a Calvi (2005) e Rizzato *et al.*(2012) que também utilizaram os mesmos procedimentos e instâncias.

As soluções obtidas neste trabalho não violaram as condições da escala de jornadas que desfavorecem o trabalho do motorista, como:

- Nenhum intervalo foi concedido com menos de 1 (uma) hora e 30 (trinta) minutos ou maior que 5 (cinco) horas;
- Nenhuma jornada teve seu tempo total maior do que 13 (treze) horas;
- Nenhuma jornada contínua de trabalho teve duração maior do que 6 (seis) horas;
- Nenhuma jornada ultrapassou a quantidade de 2 (duas) horas extras.

No início dos experimentos esperava-se que as diferenças entre os custos obtidos com a abordagem que utiliza a meta-heurística VNS aumentasse em relação aos custos obtidos com a abordagem determinística a medida que aumentasse o tamanho da instância. Mas observou-se um pico no *gap* de custos para instância AL1517, o mesmo acontecendo com o *gap* de número de jornadas entre as duas abordagens.

Conclui-se a eficiência da abordagem utilizando a meta-heurística VNS em relação à abordagem determinística para a resolução do PEM para todas as instâncias com exceção da instância de 130 viagens onde a abordagem determinística apresentou melhores resultados em relação a abordagem que utiliza a meta-heurística VNS.

Como continuidade de pesquisas relativas a este trabalho, sugere-se a ampliação da utilização do VNS com outras combinações dos procedimentos PCR e *k-swap*, aplicadas tanto para o procedimento *Shake* como para o *Local Search*. Sugere-se também a disponibilização da base de dados utilizada neste trabalho para serem utilizadas por outros pesquisadores e a adaptação do algoritmo proposto neste trabalho para ser utilizado com dados disponibilizados por outras pesquisas, para uma melhor visibilidade dos trabalhos desenvolvidos nesta área.

7 Referências

BODIN, L., GOLDEN, B., ASSAD, A., BALL, M. Routing and scheduling of vehicles and crews - The state of the art. *Computers and Operat*ions Research 10(2), 63-211 (1983)

CALVI, R. *Um Algoritmo para o Problema de Escalonamento de Tripulação em Empresas de Ônibus.* Dissertação (Mestrado em Ciências da Computação). Maringá: Universidade Estadual de Maringá, 2005. 65 p.

CAPELA, M. V.; CAPELA, J. M. V. Elaboração de Gráficos Box-Plot em Planilhas de Cálculo. *In: Anais do I Congresso de Matemática Aplicada e Computacional*, 2011, Uberlândia, MG.

CARPANETO, G.; TOTH, P. Primal-dual algorithms for the assignment problem. *Discrete Applied Mathematics*, v. 18, n. 2, p. 137–153, North-Holland, 1987.

CONSTANTINO, A. A. ; MELO, E. L. ; RIZZATO, D. B. . Um algoritmo heurístico para otimização do Problema de Escalonamento de Enfermeiros. *In: XLI Simpósio Brasileiro de Pesquisa Operacional*, 2009, Porto Seguro, BA.

DADUNA, J.R., BRANCO, L., PAIXÃO, J.M.P. Computer-Aided Transit Scheduling. *Lecture Notes in Economics and Mathematical Systems*, Springer, 1995, Berlin.

DADUNA, J.R., WREN, A. Computer-Aided Transit Scheduling, *Lecture Notes in Economics and Mathematical Systems*, Springer, 1988, Berlin.

DE LEONE, R.; FESTA, P.; MARCHITTO, E. A Bus Driver Scheduling Problem: a new mathematical model and a GRASP approximate solution. *Journal of Heuristics*, vol. 17, n. 4, p. 441-466, 2011.

DE GROOT S, HUISMAN D. Vehicle and crew scheduling: solving large real-world instances with an integrated approach. *Econometric Institute Report* EI2004-13, Erasmus University, Rotterdam, The Netherlands; p. 43-56, 2004.

DIAS, T. G.; SOUZA, J. P.; CUNHA, J. F. Genetic algorithms for the bus driver scheduling problem: a case study. *Journal of the Operational Research Society*, 53, p. 1-12, 2002.

GONÇALVES, T. L. *Meta-heurísticas para o Problema de Programação de Tripulações*. Dissertação (Mestrado em Engenharia de Sistemas e Computação), Rio de Janeiro: Universidade Federal do Rio de Janeiro, 2010. 98p.

HANSEN, P.; MLADENOVIĆ, N.; PÉREZ, J. M. Variable neighbourhood search: methods and applications. *Annals of Operational Research*, v. 175, n. 1, p. 367-407, 2010.

HILLIER, F. S.; LIEBERMAN, G. J. *Introdução à Pesquisa Operacional*. 8. ed. Porto Alegre: AMGH, 2010. 852 p.

HUISMAN, D.; FRELING, R.; WAGELMANS, A. P. M. Multiple-Depot Integrated Vehicle and Crew Scheduling. *Transportation Science*, v. 39, n. 4, p. 491–502, 2005.

KUHN, H. W. The Hungarian Method for the assignment problem. *Naval Research Logistics Quarterly*, 1955, 2:83_97.

LAURENT, B.; HAO, J. K. Simultaneous Vehicle and Crew Scheduling for Extra Urban Transports. *Lecture Notes in Artificial Intelligence* 5027: 466–475, Springer, 2008.

LOURENÇO, H. R.; PAIXÃO, J. P.; PORTUGAL, R. Multiobjective Metaheuristics for the Bus Driver Scheduling Problem. *Transportation Science*, v. 35, n. 3, p. 331-343, 2001.

MARINHO, E. H.; OCHI, L. S.; DRUMMOND, M. A.; SOUZA, M. J. F.; SILVA, G. P. Busca Tabu Aplicada ao Problema de Programação de Tripulações de Ônibus Urbano. *Anais: XXXVI SBPO*, São João Del Rey, 2004, p. 1471-1482.

MAURI, G. R. *Novas Heurísticas para o Problema de Escalonamento de Tripulações*. Dissertação (Mestrado em Computação Aplicada*)*, São José dos Campos: Instituto Nacional de Pesquisas Espaciais, 2005. 105 p.

MAURI, G. R.; LORENA, L. A. N. Método Interativo para Resolução do Problema de Escalonamento de Tripulações. *Anais: XXXVI SBPO,* 2004, São João del Rey, MG.

MELO, E. L. ; CONSTANTINO, A. A. ; RIZZATO, D. B. ; ROMÃO, W. Um algoritmo heurístico de duas fases para escalonamento de enfermeiros com balanceamento de atendimento às preferências. *Anais: XLII Simpósio Brasileiro de Pesquisa Operacional*, 2010, Bento Gonçalves. p. 1-12.

PRATA, B. A. Programação Integrada de veículos e motoristas: uma visão geral. *Revista Eletrônica Sistemas e Gestão*, v. 4, n. 3, p. 182 – 204, Set./Dez. 2010.

RIZZATO, D. B.; SAKIYAMA, R. Z.; CONSTANTINO, A. A.; ROMÃO, W. Comparação de Algoritmos Heurísticos para um Problema de Planejamento Operacional de Transporte Público. *Pré-Anais: XVI CLAIO/ XLIV SBPO, 2012*, Rio de Janeiro, RJ.

RIZZATO, D. B. ; MELO, E. L. ; CONSTANTINO, A. A. Automação e otimização do processo de escalonamento de enfermeiros. *Anais: XXX Encontro Nacional de Engenharia de Produção*, 2010, São Carlos, SP.

SANTOS, A. G.; MATEUS, G. R. Crew Scheduling Urban Problem: an Exact Column Generation Approach Improved by a Genetic Algorithm. *IEEE Congress on Evolutionary Computation*, p. 1725–1731, 2007.

SILVA, G. P.; CUNHA, C. B. Uso da técnica de busca em vizinhança de grande porte para a programação da escala de motoristas de ônibus urbano. *Revista Transportes*, v. 8 n. 2 p. 37-45, 2010.

SILVA, G. P.; SOUZA, M. J. F.; ALVES, J. M. C. B. Simulated Annealing Aplicado à Programação da Tripulação no Sistema de Transporte Público. *Anais: XXII Encontro Nacional de Engenharia de Produção,* 2002, Curitiba.

SILVA, G. P.; SOUZA, M. J. F.; REIS, J. A. Um método exato para otimizar a escala de motoristas e cobradores do sistema de transporte público. *Anais: XVIII Congresso de Pesquisa e Ensino em Transportes*, Florianópolis, SC, p. 340-346, 2004.

SOUZA, M. J. F.; RODRIGUES, M. M. S.; MAPA, S. M. S.; SILVA, G. P. Um estudo das heurísticas Simulated Annealing e VNS aplicadas ao problema de programação de tripulações. *Anais: XXIII Encontro Nacional de Engenharia de Produção*, 2003, Ouro Preto, MG.

VOβ, S., DADUNA, J.R. Computer-Aided Transit Scheduling, *Lecture Notes in Economics and Mathematical Systems*, Springer, 2011, Heidelberg.

WREN, A. Scheduling Vehicles and their Drivers – Forty Years' Experience. *Report 2004.03, Research Report Series*, University of Leeds, School of Computing Research, 2004.

WREN, A; ROUSSEAU, J.M. Bus driver scheduling – An overview, *In:Computer-Aided Transit Scheduling,* Daduna J. R., Branco, I., Paixão, J. M. P. (eds). Berlin, Gemany: Spring-Verlag, 1993, 174-187.

YUNES, T. H.; MOURA, A. V.; SOUZA, C. C. Hybrid Column Generation Approaches for Urban Transit Crew Management Problems. *Transportantion Science,* 39(2), 273-288, 2005.