HEURÍSTICA DE BUSCA EM VIZINHANÇAS VARIÁVEIS PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM MÚLTIPLAS JANELAS DE TEMPO

HUGGO SILVA FERREIRA

HEURÍSTICA DE BUSCA EM VIZINHANÇAS VARIÁVEIS PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM MÚLTIPLAS JANELAS DE TEMPO

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Thiago F. Noronha

Belo Horizonte

Novembro de 2017

Ficha catalográfica elaborada pela Biblioteca do ICEx - UFMG

Ferreira, Huggo Silva.

F383h Heurística de busca em vizinhanças variáveis para o problema de roteamento de veículos com múltiplas janelas de tempo / Huggo Silva Ferreira. – Belo Horizonte, 2017.
xvii, 53 f.: il.; 29 cm.

Dissertação (mestrado) - Universidade Federal de Minas Gerais – Departamento de Ciência da Computação.

Orientador: Thiago Ferreira de Noronha.

1. Computação – Teses. 2. Otimização combinatória 3. Programação heurística. I. Orientador. II. Título.

CDU 519.6*62(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Heurística de busca em vizinhanças variáveis para o problema de roteamento de veículos com múltiplas janelas de tempo

HUGGO SILVA FERREIRA

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

Prof. Thiago Ferreira de Noronha - Orientador
Departamento de Ciência da Computação - UFMG

Prof. Geraldo Robson Mateus
Departamento de Ciência da Computação - UFMG

Prof. Maurício Cardoso de Souza
Departamento de Engenharia de Produção - UFMG

Belo Horizonte, 30 de Novembro de 2017.

Resumo

O Problema de Roteamento de Veículos com Múltiplas Janelas de Tempo é uma generalização do Problema de Roteamento de Veículos, onde os clientes têm uma ou mais janelas de tempo nas quais eles podem ser visitados. A melhor heurística na literatura, HVNTS, é uma hibridização das metaheurísticas *Tabu Search* e *Variable Neighborhood Search* que trabalha principalmente com soluções inviáveis, pois assume-se que pode ser necessário passar por soluções inviáveis para chegar em diferentes regiões do espaço de busca. Nesta dissertação, propomos uma heurística de *Variable Neighborhood Search* mais simples, onde todo o esforço computacional é gasto na busca de soluções viáveis. Experimentos computacionais mostraram que a heurística proposta é competitiva com a melhor heurística na literatura.

Palavras-chave: Busca em vizinhanças variáveis, Problema de roteamento de veículos, Múltiplas Janelas de Tempo.

Abstract

The Vehicle Routing Problem (VRP) with Multiple Time Windows is a generalization of VRP, where the customers have one or more time windows in which they can be visited. The best heuristic in the literature, HVNTS, is a hybridization of the Tabu Search and Variable Neighborhood Search metaheuristics that mostly deals with infeasible solutions, because it is assumed that one may not reach some regions of the search space without passing through infeasible solutions. In this dissertation, we propose a simpler Variable Neighborhood Search heuristic where all the computational effort is spent on searching for feasible solutions. Computational experiments showed that the proposed heuristic is competitive with the best heuristic in the literature.

Keywords: Variable Neighborhood Search, Vehicle Routing Problem, Multiple Time Windows.

Lista de Figuras

Lista de Tabelas

List of Algorithms

Sumário

Capítulo 1

Introdução

A otimização no campo de roteamento de veículos é impactante em situações reais, de forma que criar rotas mais econômicas é de interesse tanto econômico quanto ambiental e social. Em relação ao interesse econômico, conforme CNT [2016], 12,7% do PIB brasileiro foi destinado a custos logísticos. Desses 12,7%, 6,8% do PIB, aproximadamente R$ 401 bilhões, foram gastos apenas com transporte. Além disso, Alvarenga [2005] destaca que o custo de transporte é responsável por de 10% a 15% do valor final de uma mercadoria. Com base nisso, a partir de um melhor planejamento de rotas, gastos com transporte e problemas relacionados à transporte (como a manutenção de ruas municipais ou rodovias) podem ser minimizados, levando a minimização de preços de mercadorias, o que é interessante tanto para empresas quanto para consumidores finais.

Quanto ao impacto ambiental, o setor de transporte é responsável por 8.7% das emissões de CO_2, sendo só o rodoviário responsável por 7.8% [MINC, 2013]. Também é destacado que, no Brasil, os combustíveis mais consumidos foram o diesel (47.4%) e a gasolina (39.1%), sendo que, em transporte, sua maior parte foi devido ao transporte rodoviário. Conforme MINC [2013], em 2012, 49% da emissão de CO_2 pelo setor de transporte veio da queima de óleo diesel e 33% de gasolina. Portanto, a partir desses dados, a pesquisa em roteamento de veículos pode ter um impacto ambiental significativo, pois quanto menor a rota gerada para um veículo, menor a emissão de gases causadores de efeito estufa por parte desse veículo.

Vehicle Routing Problem (VRP) foi introduzido pela primeira vez por Dantzig & Ramser [1959]. VRP é um problema de otimização combinatória que consiste em definir uma configuração de rotas para uma frota de veículos de modo a suprir um conjunto de demandas de clientes. Quando possui apenas a restrição de capacidade do veículo, o problema é conhecido como *Capacited Vehicle Routing Problem* (CVRP)

[Toth & Vigo, 2014].

VRP possui diversas aplicações em situações reais como coleta e entrega de bebidas e alimentos e diversos outros produtos [Golden et al., 2001]. Porém, características de algumas situações reais fazem com que o VRP não seja totalmente adequado para modela-las, o que gera a necessidade de criação de variações de VRP mais adequadas a essas situações. Pode-se citar como exemplos dessas variações: *Multiple Depot Vehicle Routing Problem* (MDVRP) [Renaud et al., 1996], que modela situações em que as entregas podem partir de vários depósitos, *Vehicle Routing Problem with Pickup and Delivery* (VRPPD) [Savelsbergh & Sol, 1995], em que o conjunto de demandas é dividido em coletas e entregas, sendo que, para uma determinada entrega, antes deve ser feita sua respectiva coleta, e *Vehicle Routing Problem with Time Windows* (VRPTW) [Savelsbergh, 1985], em que cada cliente deve ser atendido dentro de um intervalo de tempo pré-estabelecido, isso é, uma janela de tempo. Todas essas variações são generalizações de VRP e, portanto, também são NP-Difíceis [Raff, 1983], [Berbeglia et al., 2007] e [Solomon & Desrosiers, 1988].

Em VRPTW, cada cliente possui uma única janela de tempo. Porém, em várias aplicações reais que envolvem roteamento, é comum que um cliente possua mais de uma janela de tempo, como, por exemplo, no planejamento de rotas para representantes de vendas [Tricoire et al., 2010]. Portanto, para modelar essas situações, foi proposto o *Vehicle Routing Problem with Multiple Time Windows* (VRPMTW) [de Jong et al., 1996].

VRPMTW pode ser definido da seguinte forma. Considere um conjunto de veículos com a mesma capacidade e o mesmo tempo máximo para a duração de suas rotas, e um conjunto de clientes, onde cada um tem uma demanda, um tempo de serviço e um conjunto de janelas de tempo. O problema consiste em definir um conjunto de rotas que começam e terminam em um mesmo depósito. Nessa rota, os clientes são são visitados apenas uma vez e seus tempos de serviço começam em qualquer uma de suas respectivas janelas de tempo. O objetivo é minimizar uma função que combina o tamanho da frota utilizada, isso é, o número de veículos cuja rota não é vazia, e o somatório do tempo que cada veículo levou para visitar todos os clientes e retornar ao depósito [Belhaiza et al., 2014].

VRPMTW é formalmente definido da seguinte forma. Seja $G = (V, A)$ um grafo completo, onde $V = \{0\} \cup N$ é o conjunto de nós, em que 0 representa o depósito e $N = \{1, 2, ..., |N|\}$ representa os clientes; e A é o conjunto de arestas, com cada aresta $[i, j] \in A$ associada a um tempo de viagem t_{ij}. Cada nó $i \in N$ está associado a uma demanda q_i, um tempo de serviço s_i e um conjunto de janelas de tempo J_i, em que cada janela $[a, b] \in J_i$ define um período de tempo em que o serviço do cliente i pode

começar. Além disso, seja R um conjunto de veículos, em que cada veículo $k \in R$ está associado a uma capacidade Q e uma duração máxima de rota D. É assumido que $|R| = |N|$, isso é, o número de veículos é maior ou igual ao número de clientes. Cada cliente $i \in N$ precisa ser visitado apenas uma vez por apenas um veículo $k \in R$. Se um veículo k chega em um cliente i fora de uma de suas janelas de tempo, ele deve esperar um tempo w_i^k até o começo da próxima janela de tempo de i. Além disso, a soma de todas as demandas dos clientes visitados na rota do veículo $k \in R$ não pode ser maior do que a capacidade do veículo Q. Mais ainda, as rotas devem começar e terminar no nó 0, e a soma do tempo de viagem, do tempo de espera e do tempo de serviço de todos os clientes visitados na rota do veículo $k \in R$ não pode ser maior do que a duração máxima de rota D.

Nos trabalhos sobre VRPMTW, duas funções objetivo são utilizadas. A função objetivo proposta em [Favaretto et al., 2007] e usada em [Belhaiza et al., 2014] define o custo de uma solução como a soma de três valores: (i) o tempo total de viagem; (ii) o tempo de espera em cada cliente; (iii) o número de veículos utilizados multiplicado por uma constante F que representa o custo de utilizar um veículo em unidades de tempo. Essa função objetivo é estabelecida na Equação (1.1), onde $x_{ij}^k = 1$ se o arco (i, j) está na rota do veículo $k \in R$. Além disso, $r^k = 1$ se a rota do veículo $k \in R$ não é vazia, caso contrário $r^k = 0$ e w_i^k é o tempo de espera do veículo k no cliente i.

$$f(x) = \sum_{k \in R} \sum_{i \in V, j \in V \setminus \{i\}} t_{ij} x_{ij}^k + \sum_{k \in R} \sum_{i \in V} w_i^k + F \sum_{k \in R} r^k \tag{1.1}$$

A segunda função objetivo utilizada foi proposta em [Belhaiza et al., 2014]. Assim como (1.1), ela minimiza o tempo total de viagem e o número de veículos utilizado. Entretanto, neste caso, o tempo de espera em cada cliente é desconsiderado. Essa função objetivo é descrita na Equação (1.2). Esta dissertação foca nessas duas variações de VRPMTW.

$$f(x) = \sum_{k \in R} \sum_{i \in V, j \in V \setminus \{i\}} t_{ij} x_{ij}^k + F \sum_{k \in R} r^k \tag{1.2}$$

O VRPMTW é formulado como um problema de programação linear inteira por Belhaiza et al. [2014] da seguinte forma. Além das constantes e variáveis definidas anteriormente, considere as variáveis binárias z_i^k, que indica se o cliente i está na rota do veículo k, e v_i^p, que é igual a 1 quando o cliente i é atendido dentro de sua $p-$ésima janela de tempo. Sejam também, y_{ij}^k o fluxo no arco x_{ij}^k, l_i^k o tempo de chegada do veículo k no cliente i e m_i^k o tempo de saída do veículo k no cliente i. Por fim, seja M

uma constante arbitrariamente grande. Então, temos:

Minimizar

$$f(x) = \sum_{k \in R} \sum_{i \in V, j \in V \setminus \{i\}} t_{ij} x_{ij}^k + \sum_{k \in R} \sum_{i \in V} w_i^k + F \sum_{k \in R} r^k$$

ou

$$f(x) = \sum_{k \in R} \sum_{i \in V, j \in V \setminus \{i\}} t_{ij} x_{ij}^k + F \sum_{k \in R} r^k$$

sujeito a

$$\sum_{k \in R} z_i^k = 1, \ i \in V, \tag{1.3}$$

$$\sum_{j \in V} x_{ji}^k = \sum_{j \in V} x_{ij}^k, \ i \in V \text{ and } k \in R, \tag{1.4}$$

$$2x_{ij}^k \leq z_i^k + z_j^k, \ i, \ j \in V \text{ and } k \in R, \tag{1.5}$$

$$\sum_{k \in R} \sum_{j \in V} x_{ij}^k \leq 1, \ i \in V \text{ and } k \in R, \tag{1.6}$$

$$\sum_{k \in R} \sum_{j \in V} x_{ji}^k \leq 1, \ i \in V \text{ and } k \in R, \tag{1.7}$$

$$y_{ij}^k \leq Q x_{ij}^k, \ i, \ j \in V \text{ and } k \in R, \tag{1.8}$$

$$\sum_{j \in V} y_{ji}^k - \sum_{j \in V} y_{ij}^k \geq q_i z_i^k, \ i \in V \text{ and } k \in R, \tag{1.9}$$

$$m_0^k \geq -M(1 - z_0^k), \ k \in R, \tag{1.10}$$

$$l_0^k \leq D + M(1 - z_0^k), \ k \in R, \tag{1.11}$$

$$l_0^k - m_0^k \leq D + M(1 - z_0^k), \ k \in R, \tag{1.12}$$

$$m_i^k \geq l_i^k + w_i^k + s_i - M(1 - z_i^k), \ i \in N \text{ and } k \in R, \tag{1.13}$$

$$l_j^k \geq m_i^k + t_{ij} - M(1 - x_{ij}^k), \ i, \ j \in V \text{ and } k \in R, \tag{1.14}$$

$$l_j^k \leq m_i^k + t_{ij} - M(1 - x_{ij}^k), \ i, \ j \in V \text{ and } k \in R, \tag{1.15}$$

$$l_i^k + w_i^k \geq a_i^p - M(1 - z_i^k) - M(1 - v_i^p), \ i \in N, \ p \in J_i \text{ and } k \in R, \tag{1.16}$$

$$l_i^k + w_i^k \leq b_i^p - M(1 - z_i^k) - M(1 - v_i^p), \ i \in N, \ p \in J_i \text{ and } k \in R, \tag{1.17}$$

$$\sum_{p=1}^{|J_i|} v_i^p = 1,\ i \in N, \tag{1.18}$$

$$r_i^k \geq z_i^k,\ i \in V \text{ and } k \in R, \tag{1.19}$$

$$y_{ij}^k,\ w_i^k,\ l_i^k,\ m_i^k \geq 0, \tag{1.20}$$

$$r^k,\ x_{ij}^k,\ v_i^k,\ z_i^k \text{ são binários.} \tag{1.21}$$

As restrições (1.3) afirmam que cada cliente é designado para apenas um veículo. As restrições (1.4) definem que as rotas iniciam e terminam no depósito e que o número de veículos saindo de um cliente i seja o mesmo que o número de veículos chegando no mesmo cliente. As restrições (1.5) definem que um arco x_{ij}^k só pode ser atravessado pelo veículo k se z_i^k e z_j^k forem iguais a um. As restrições (1.6) e (1.7) fazem com que cada cliente i seja visitado por apenas um veículo. As restrições (1.8) fazem com que fluxo do arco $(i,\ j)$ seja limitado pela capacidade Q do veículo. As restrições (1.9) definem que a demanda de cada cliente i presente na rota do veículo k seja satisfeita. As restrições (1.10) garantem que a saída de um veículo k do depósito não seja menor que zero e as restrições (1.11) fazem com que a chegada do veículo k no depósito seja superior a D. As restrições (1.12) garantem que a duração de uma rota também não ultrapasse o limite máximo de duração D de uma rota.

As restrições (1.13) garantem que o tempo de saída de um veículo K de um cliente i seja, pelo menos, igual ao tempo de chegada no cliente i, mais o tempo de espera nesse cliente e mais o tempo de serviço no cliente caso ele seja atribuído ao veículo k. As restrições (1.14) e (1.15) garantem que o tempo de chegada no cliente j seja igual ao tempo de saída do cliente i, mais o custo de viagem t_{ij} apenas se esse arco fizer parte da rota do veículo k. As restrições (1.16) e (1.17) definem que o tempo de chegada no cliente i mais o tempo de espera nesse cliente pelo veículo k esteja dentro da p-ésima janela de tempo escolhida. As restrições (1.18) definem que apenas uma janela de tempo seja escolhida por cliente. As restrições (1.19) garantem que um cliente i seja atribuído ao veículo k se esse veículo é utilizado. Finalmente, as restrições (1.20) e (1.21) definem os intervalos factíveis para as variáveis de decisão.

A melhor heurística na literatura para essas duas variações de VRPMTW é chamda de *Hybrid Variable Neighborhood Tabu Search* (HVNTS) [Belhaiza et al., 2014]. Ela é uma hibridização das metaheurísticas *Variable Neighborhood Search* (VNS) [Mladenović & Hansen, 1997] e *Tabu Search* [Glover, 1986]. Essa heurística é proposta a partir da conjectura de que é necessário passar por regiões inviáveis do espaço de busca para se encontrar soluções de boa qualidade. Sendo assim, as restrições, relativas a (i) capacidade do veículo, (ii) janelas de tempo e (iii) duração máxima de rota são

relaxadas, e adicionadas como penalidades fixas na função objetivo nessa heurística. Estas penalidades devem ser suficientemente grandes para que soluções inviáveis sejam permitidas, mas desencorajadas se existirem soluções viáveis semelhantes. Embora as soluções viáveis não sejam garantidas, os experimentos computacionais mostraram que foram encontradas soluções viáveis para todas as instâncias testadas.

Nesta dissertação, é proposta uma heurística de VNS que apenas trabalha com soluções viáveis. Portanto, nenhum esforço computacional é gasto em soluções inviáveis. Além disso, com esta heurística, pretende-se mostrar que diferentes regiões do espaço de busca, onde são encontradas boas soluções, podem ser alcançadas sem passar por soluções inviáveis.

O restante deste trabalho está organizado da seguinte forma. Trabalhos relacionados são analizados na revisão bibliográfica apresentada na Seção 2. A heurística VNS proposta é discutida na Seção 3. Experimentos computacionais que comparam os resultados da heurística VNS proposta com os de HVNTS são relatados na Seção 4. As observações finais são feitas e os trabalhos futuros são discutidos na última seção.

Capítulo 2

Revisão Bibliográfica

Neste capítulo, são apresentados trabalhos relacionados a VRPMTW na literatura. Primeiramente, são apresentados trabalhos que sintetizam resultados importantes sobre VRP e VRPTW. Em seguida, são discutidos trabalhos sobre problemas de otimização combinatória com multiplas janelas de tempo. Por fim, são apresentados trabalhos na literatura sobre, especificamente, VRPMTW.

Laporte [2007] apresenta um estudo sobre os diversos tipos de técnicas utilizadas na resolução de VRP. Quanto a algoritmos exatos, o autor foca em formulações para programação linear inteira, sendo elas a *Two-Index Vehicle Flow* [Laporte et al., 1985], *Two-Index Two-Commodity Flow* [Baldacci et al., 2004] e *Set-Partitioning* [Balinski & Quandt, 1964]. Já sobre heurísticas e metaheurísticas para o problema, Laporte [2007] destaca diversas heurísticas contrutivas, buscas locais e metaheurísticas e dá detalhes de suas implementações para VRP. Nesse trabalho, conclui-se por dizer que os algoritmos exatos, todos baseados no método *branch-and-cut*, conseguem resolver instâncias até 100 clientes. Dentre as metaheurísticas, os melhores resultados foram obtidos por aquelas que utilizam busca local, algoritmos genéticos [Holland, 1992] ou uma combinação de ambos.

Para VRPTW, Kallehauge [2008] faz um estudo sobre formulações derivadas daquelas propostas para TSP e algoritmos para VRPTW desenvolvidos com base nessas formulações. O autor observa que os métodos para VRPTW são, em sua maioria, herdados de formulações para TSP. Portanto, o autor dá atenção a quatro formulações de TSP e lista alguns artigos que utilizam dessas variações para criar algoritmos para VRPTW. São elas: *Arc formulation* [Dantzig et al., 1954], *Arc-node formulation* [Miller et al., 1960], *Spanning tree formulation* [Held & Karp, 1970] e [Held & Karp, 1971] e *Path formulation* [David Houck, 1978]. Nesse trabalho, conclui-se que os algoritmos exatos de melhor desempenho são aqueles baseados em *path formulation*.

Solomon [1987] fez um estudo entre seis heurísticas construtivas para comparar seus desempenhos em VRPTW. As heurísticas utilizadas foram a *Saving Heuristic*, *Time Oriented Nearest Neighbor Heuristic*, *Time Sweep Heuristic* e três heurísticas de inserção de sua própria autoria. Dessas heurísticas, *Push Foward Insertion Heuristic* (PFIH) foi a que obteve os melhores resultados na maioria das instâncias testadas. Essa heurística considera a inserção de um cliente levando em consideração o custo incremental de adicioná-lo em uma determinada posição. Ela constrói uma solução com base em dois critérios. O primeiro critério consiste de selecionar o melhor cliente a ser inserido na iteração corrente e o segundo consiste de decidir qual a melhor posição em que esse cliente deve ser inserido. No caso de não existir uma posição viável para inserção de um cliente, uma nova rota é criada para atendê-lo.

Bräysy & Gendreau [2005a] fizeram um estudo sobre heurísticas construtivas e de busca local para VRPTW. Das heurísticas construtivas estudadas, a proposta por Ioannou et al. [2001] foi a que obteve melhores soluções, mas é uma das mais lentas dentre as estudadas. Essa heurística é baseada na heurística PFIH, porém ela utiliza um critério para seleção baseado em análise antecipatória. Já dentre as heurísticas de busca local, a heurística proposta por Schrimpf et al. [2000] foi a que obteve melhor desempenho. A heurística consiste em uma estratégia onde um conjunto de clientes é retirado de uma solução e, logo em seguida, reinserido conforme um critério guloso. Os autores concluem que heurísticas híbridas, isso é, aquelas feitas pela combinação de mais de uma metaheurística, têm melhor desempenho em termos de qualidade de solução.

Bräysy & Gendreau [2005b] também fizeram uma revisão sobre metaheurísticas para VRPTW. Para comparação, foi seguido o mesmo método utilizado para as heurísticas construtivas e de busca local. Os autores concluem que, dentre as heurísticas estudadas, as de melhor robustez foram as propostas em [Bent & Van Hentenryck, 2004], [Bräysy, 2003], [Ibaraki et al., 2005], [Berger et al., 2003] e [Homberger & Gehring, 2005]. A heurística proposta por Bent & Van Hentenryck [2004] é baseada em *Simulated Annealing* [Kirkpatrick et al., 1983] e *Very Large Neighborhood Search* [Ahuja et al., 2000]. A heurística proposta por Bräysy [2003] é baseada em VNS. A heurística porposta por Ibaraki et al. [2005] é baseada nas metaheurísticas *Iterated Local Search* (ILS) [Lourenço et al., 2003] e *Greedy Randomized Adaptive Search Procedure* (GRASP) [Feo & Resende, 1989]. As heurísticas propostas por Berger et al. [2003] e por Homberger & Gehring [2005] são baseadas em Algoritmos Genéticos [Holland, 1992].

Nagata et al. [2010] propõem em seu artigo um algoritmo memético para a resolução de VRPTW. As soluções iniciais são geradas através da heurística RM [Nagata &

Bräysy, 2009], que consiste em criar um conjunto de clientes não atendidos e, a partir dele, inserir os clientes em rotas. Caso o número de rotas na solução não seja igual a um limite inferior para o número mínimo de rotas, uma das rotas é desfeita e seus clientes são reinseridos no conjunto de clientes não atendidos. Se, para a inserção de um cliente, é necessário remover outro, esse outro cliente é removido da rota e inserido no conjunto de clientes não atendidos. O processo se repete até que o limite inferior para o número mínimo de rotas seja atingido ou outro critério de parada seja atingido, como um número máximo de iterações. Os autores concluem que o algoritmo foi capaz de melhorar 184 das 300 das melhores soluções conhecidas para as instâncias de Gehring & Homberger [1999].

Dentre os trabalhos sobre problemas de otimização combinatória com múltiplas janelas de tempo, pode-se citar o trabalho de Christiansen & Fagerholt [2002] sobre o problema de *Ship Scheduling with Multiple Time Windows* (SSMTW). Esse problema se caracteriza por planejar rotas de navios entre portos de carregamento e de entrega de forma que a distância entre cada parada seja minimizada. As janelas de tempo tratadas no problema são relativas aos períodos em que um porto está aberto para carga e descarga durante a semana. SSMTW foi tratado em [Christiansen & Fagerholt, 2002] com uma abordagem de particionamento de conjuntos. Primeiro, foram gerados todas as programações viáveis para cada porto utilizando de um algoritmo de programação dinâmica, mas sem considerar as janelas de tempo. Em seguida, foi utilizada uma função de penalidade nos tempos de serviço mais próximos do fim de semana. As programações viáveis foram, então, utilizadas em um modelo de particionamento de conjuntos e resolvidas por um software comercial de programação linear inteira.

Pesant et al. [1999] apresentam em seu trabalho um algoritmo para *Traveling Salesman Problem with Multiple Time Windows* (TSPMTW). Esse algoritmo é proposto por adaptar, para TSPMTW, um algoritmo exato proposto em [Pesant et al., 1998] para *Traveling Salesman Problem with Time Windows* (TSPTW). Isso foi feito por adaptar o modelo usado no algoritmo para TSPMTW e usar uma estratégia de *aprofundamento de custo interativo* [Pesant et al., 1998].

Tricoire et al. [2010] propuseram uma heurística baseada em *Variable Neighborhood Descent* (VND) [Mladenović & Hansen, 1997] para *Multi-Period Orienteering Problem with Multiple Time Windows* (MuPOPTW) [Tricoire et al., 2010]. Nesse problema, os clientes estão divididos em obrigatórios e opcionais e possuem uma ou duas janelas de tempo para cada dia do horizonte de planejamento. PFIH [Solomon, 1987] é utilizada para construir uma solução inicial e, em seguida, é executado um VND, desconsiderando as restrições de janelas de tempo, o que pode levar a soluções inviáveis. Para viabilizar uma rota, é proposto um algoritmo exato que seleciona para cada

cliente o período em que ele deve ser atendido dentro de uma de suas janelas de tempo. Isso é feito de forma a minimizar o tempo total da rota e a manter a viabilidade das soluções. Caso não seja possível viabilizar a solução, ela é descartada por ser inviável.

O VRPMTW foi proposto em [de Jong et al., 1996]. Nesse trabalho, os autores propõem uma formulação para VRPMTW e discutem as diferenças no tratamento de VRPMTW em relação a VRPTW. Os autores também propõem procedimentos para definir a duração de uma rota. Eles concluem que o uso desses procedimentos levam a criação de melhores soluções por heurísticas construtivas.

Em [Favaretto et al., 2007] é proposta uma formulação de Programação Linear Inteira (PLI) para VRPMTW que não foi capaz de resolver instâncias de médio e grande porte utilizadas nos experimentos computacionais. Portanto, os autores propõem uma heurística chamada de *Multiple Ant Colony System* (MACS). A heurística começa por construir uma solução utilizando uma heurística baseada em *Nearest Neighbor* [Lin & Kernighan, 1973] e, em seguida, duas colônias de formigas trabalham sobre essa solução. A primeira tem como objetivo minimizar o número de rotas criadas e a segunda, utilizando-se da solução fornecida pela primeira, procura reduzir o tamanho total das rotas. Experimentos computacionais mostraram que os tempos de execução dessa heurística aumentam significativamente com a quantidade de janelas de tempo.

Belhaiza et al. [2014] propuseram a heurística chamada de *Hybrid Variable Neighborhood Tabu Search* (HVNTS), na qual utilizaram uma lista tabu para melhorar o desempenho de uma heurística de VNS. A heurística permite a criação de soluções inviáveis desde que uma função de penalidade seja utilizada. Ela utiliza oito operadores para a busca local e cinco para a perturbação. Os oito operadores de busca local, detalhados no Capítulo 3, são subdivididos entre operadores de rota única e operadores de múltiplas rotas. Os operadores de rota única são o *single-route relocate*, *single-route 2-exchange* e *single-route invert*. Já os de rotas múltiplas são o *multi-route relocate*, *multi-route swap*, *multi-route cross*, *multi-route 3-node swap* e *multi-route 3-exchange*. Os cinco operadores de vizinhança que foram utilizados na perturbação são uma variação do *single-route 2-exchange*, uma variação do *multi-route swap* e três variações do *multi-route cross*.

Como HVNTS trabalha com soluções inviáveis, Belhaiza et al. [2014] propõem um algoritmo para redução do tempo de espera em uma rota e, possivelmente, viabilizar a solução. Esse algoritmo, chamado de *Minimum Backtrack Slack Algorithm* (MBTSA), atrasa a saída de um veículo do depósito de forma a minimizar o tempo de espera na rota. Para tanto, dada uma rota, o algoritmo começa por definir o tempo de chegada em cada cliente considerando apenas o tempo de viagem entre ele e o cliente que o precede. As restrições de janela de tempo são desconsideradas nessa fase. Em seguida,

o algoritmo define o atraso mínimo e máximo da saída do veículo do depósito admitido para cada janela de tempo de cada cliente na rota. Isso é feito por subtrair o tempo de chegada da abertura e do fechamento de cada janela de tempo. Se uma janela de tempo fechar antes do tempo de chegada, ela não é mais considerada nas fases subsequentes. Por fim, MBTSA computa o atraso da saída do depósito que resulta no menor custo da solução. Para isso, ele considera todas as janelas de tempo de todos os clientes de uma rota e retorna a melhor configuração em que é minimizado o tempo de espera. A complexidade desse algoritmo é $O(\sum_{k \in R} \prod_{i=1}^{size(k,S)} |J_i|)$, onde $|J_i|$ é o número de janelas de tempo de um cliente i e $size(k, S)$ é o número de clientes na rota com mais clientes na solução. Os autores observam que o HVNTS foi capaz de superar o desempenho da heurística MACS proposta por Favaretto et al. [2007] e, portanto, é a melhor heurística para VRPMTW na literatura.

Capítulo 3

Heurística de Busca em Vizinhanças Variáveis para VRPMTW

A heurística proposta nesta dissertação, chamada de VNS-VRPMTW, é composta por três heurísticas que são executadas em sequência. Primeiro, a heurística construtiva, baseada na heurística PFIH, cria uma solução inicial viável. Em seguida, a heurística *Route-Elimination* é executada para reduzir o número de rotas de veículos não vazias nesta solução. Por fim, a solução resultante é otimizada pela heurística *Route-Optimization*, baseada em VNS, que minimiza o custo total da função objetivo do problema. Essas três heurísticas, assim como os operadores de vizinhança que elas utilizam, são descritas a seguir.

3.1 Operadores de Vizinhança

Os operadores de vizinhança utilizados na heurística proposta nesta dissertação são um subconjunto daqueles usados na heurística HVNTS [Belhaiza et al., 2014]. Eles são divididos em *operadores de rota única* e *operadores de múltiplas rotas*. Em todas as etapas deste trabalho, somente operações que resultam em vizinhos viáveis são executadas. Nesta seção, denota-se por S^k o conjunto de arcos (i, j), onde $i \in V$ e $j \in V$, na rota de um veículo $k \in R$ em uma solução S. Por exemplo, considere a seguinte rota de um veículo k com cinco clientes $\langle 0, 1, 2, 3, 4, 5, 0 \rangle$. Nesse caso, o conjunto de arcos da rota do veículo k é, portanto, $S^k = \{(0, 1), (1, 2), (2, 3), (3, 4), (4, 5), (5, 0)\}$. Esse exemplo é ilustrado na Figura 3.1, onde o depósito é representado por um quadrado preto.

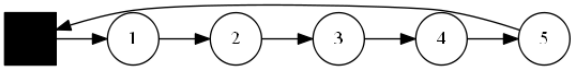

Figura 3.1. Exemplo de uma rota $\langle 0, 1, 2, 3, 4, 5, 0\rangle$ com as arestas $S^k = \{(0, 1), (1, 2), (2, 3), (3, 4), (4, 5), (5, 0)\}$.

Belhaiza et al. [2014] utilizou três operadores de vizinhança de rota única. Eles são chamados de *single-route relocate*, *single-route 2-exchange* e *single-route invert*. Esses operadores consistem em fazer alterações na posição de um ou mais clientes de uma mesma rota e são descritos a seguir.

Seja S uma solução de VRPMTW, onde S^k denota a rota do veículo $k \in R$ e $S^k = \emptyset$ se o veículo k não é utilizado na solução S. O operador *single-route relocate* consiste em remover um cliente da sua posição atual e inseri-lo em outra posição na mesma rota. Logo, dados um cliente $v \in V$ na rota do veículo $k \in R$ e um arco (i, j) $\in S^k : i \neq v, j \neq v$, um novo vizinho de S pode ser obtido removendo-se os arcos (u, v) $\in S^k$, (v, w) $\in S^k$ e (i, j) $\in S^k$ e inserindo os arcos (i, v), (v, j) e (u, w) em S^k. Por exemplo, a Figura 3.1 mostra a rota $\langle 0, 1, 2, 3, 4, 5, 0\rangle$ de um veículo k, onde $S^k = \{(0, 1), (1, 2), (2, 3), (3, 4), (4, 5), (5, 0)\}$. Já a Figura 3.2 ilustra a nova rota $\langle 0, 1, 2, 4, 3, 5, 0\rangle$ que pode ser obtida através de uma operação de *single-route relocate* do cliente 4 para a posição entre os clientes 2 e 3. Os arcos pontilhados representam os arcos eliminados da rota.

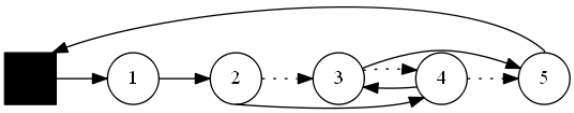

Figura 3.2. Exemplo de rota resultante do operador *single-route relocate*.

O operador *single-route 2-exchange* consiste em trocar a posição de quaisquer dois clientes em uma única rota. Logo, dados dois clientes v e $d \in V$ na rota do veículo $k \in R$, um novo vizinho de S pode ser obtido removendo-se os arcos (u, v) $\in S^k$, (v, w) $\in S^k$, (i, d) $\in S^k$ e (d, j) $\in S^k$ e inserindo os arcos (u, d), (d, w), (i, v) e (v, j) em S^k. A Figura 3.3 ilustra a rota $\langle 0, 1, 4, 3, 2, 5, 0\rangle$ que pode ser obtida através de uma operação de *single-route 2-exchange* dos clientes 2 e 4 na rota $\langle 0, 1, 2, 3, 4, 5, 0\rangle$. Os arcos pontilhados representam os arcos eliminados da rota.

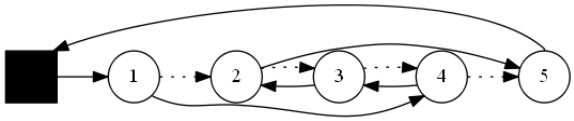

Figura 3.3. Exemplo de rota resultante do operador *single-route 2-exchange*.

O operador *single-route invert* consiste em inverter as posições de uma sequência de clientes consecutivos em uma rota. Logo, nesse caso, o último cliente dessa sequência é atendido primeiro, seguido pelo anterior até que o primeiro cliente dessa sequência seja atendido. Esse operador, mesmo sendo usado em [Belhaiza et al., 2014], não foi utilizado neste trabalho, pois a inversão de uma sequência de clientes tende a gerar soluções inviáveis devido às janelas de tempo.

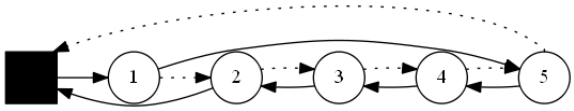

Figura 3.4. Exemplo de rota resultante do operador *single-route invert*.

Também em [Belhaiza et al., 2014], foram utilizados cinco operadores de vizinhança de múltiplas rotas, sendo eles o *multi-route relocate*, *multi-route swap*, *multi-route cross*, *multi-route 3-node swap* e *multi-route 3-exchange*. Esses operadores consistem em fazer alterações na posição de um ou mais clientes de duas ou três rotas. Eles são descritos a seguir.

O operador *multi-route relocate* consiste em remover um cliente da sua posição atual em uma rota e inseri-lo em outra posição em outra rota que não a original. Logo, dados um cliente $v_1 \in V$ na rota do veículo $k_1 \in R$ e um arco $(i_2, j_2) \in S^{k_2}$ de um veículo $k_2 \in R$, um novo vizinho de S pode ser obtido removendo-se os arcos $(u_1, v_1) \in S^{k_1}$, $(v_1, w_1) \in S^{k_1}$ e $(i_2, j_2) \in S^{k_2}$ e inserindo os arcos (i_2, v_1) e (v_1, j_2) em S^{k_2} e (u_1, w_1) em S^{k_1}. A Figura 3.5 ilustra as rotas $\langle 0, 1, 2, 4, 5, 0\rangle$ e $\langle 0, 6, 7, 3, 8, 9, 10, 0\rangle$ dos veículos k_1 e k_2, respectivamente, que podem ser obtidas através de uma operação de *multi-route relocate* do cliente 3 para a posição entre os clientes 7 e 8 nas rotas $\langle 0, 1, 2, 3, 4, 5, 0\rangle$ e $\langle 0, 6, 7, 8, 9, 10, 0\rangle$. Os arcos pontilhados representam os arcos eliminados das rotas.

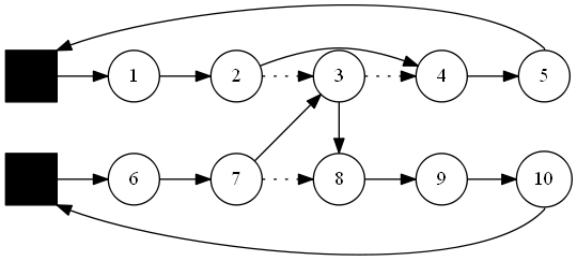

Figura 3.5. Exemplo de rotas resultantes do operador *multi-route relocate*.

O operador *multi-route swap* consiste em trocar a posição de dois clientes de duas rotas distintas. Logo, dados dois clientes $v_1 \in V$ na rota do veículo $k_1 \in R$ e $v_2 \in V$ na rota do veículo $k_2 \in R$, um novo vizinho de S pode ser obtido removendo-se os arcos $(u_1,\ v_1) \in S^{k_1}$, $(v_1,\ w_1) \in S^{k_1}$, $(u_2,\ v_2) \in S^{k_2}$ e $(v_2,\ w_2) \in S^{k_2}$ e inserindo os arcos $(u_2,\ v_1)$ e $(v_1,\ w_2)$ em S^{k_2} e $(u_1,\ v_2)$ e $(v_2,\ w_1)$ em S^{k_1}. A Figura 3.6 ilustra as rotas $\langle 0, 1, 2, 8, 4, 5, 0\rangle$ e $\langle 0, 6, 7, 3, 9, 10, 0\rangle$ dos veículo k_1 e k_2, respectivamente, que podem ser obtidas através de uma operação de *multi-route swap* dos clientes 3 e 8 nas rotas $\langle 0, 1, 2, 3, 4, 5, 0\rangle$ e $\langle 0, 6, 7, 8, 9, 10, 0\rangle$. Os arcos pontilhados representam os arcos eliminados das rotas.

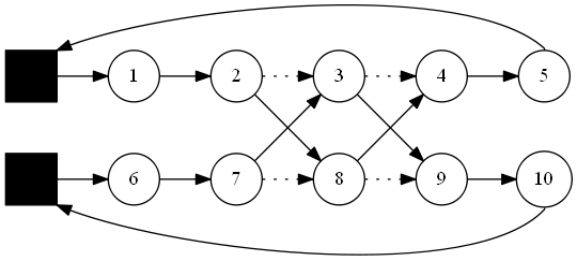

Figura 3.6. Exemplo de rotas resultantes do operador *multi-route swap*.

O operador *multi-route cross* consiste em trocar as posições de dois pares de clientes consecutivos entre duas rotas. Logo, dados dois arcos $(v_1,\ d_1) \in S^{k_1}$ e $(v_2,\ d_2) \in S^{k_2}$, um novo vizinho de S pode ser obtido removendo-se os arcos $(u_1,\ v_1) \in S^{k_1}$, $(d_1,\ w_1) \in S^{k_1}$, $(u_2,\ v_2) \in S^{k_2}$ e $(d_2,\ w_2) \in S^{k_2}$ e inserindo os arcos $(u_2,\ v_1)$ e $(d_1,\ w_2)$ em S^{k_2} e $(u_1,\ v_2)$ e $(d_2,\ w_1)$ em S^{k_1}. A Figura 3.7 ilustra as rotas $\langle 0, 1, 2, 8, 9, 5, 0\rangle$ e $\langle 0, 6, 7, 3, 4, 10, 0\rangle$ dos veículos k_1 e k_2, respectivamente, que podem ser obtidas através de uma operação de *multi-route cross* dos arcos (3, 4) e (8, 9) nas rotas $\langle 0, 1, 2, 3, 4, 5, 0\rangle$ e $\langle 0, 6, 7, 8, 9, 10, 0\rangle$. Os arcos pontilhados representam os arcos eliminados das rotas.

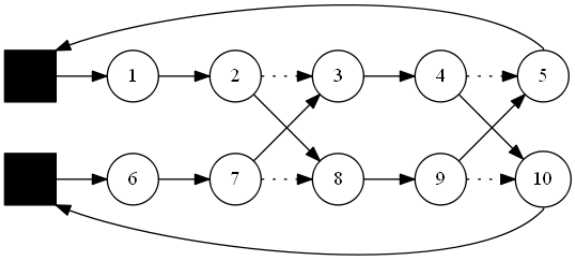

Figura 3.7. Exemplo de rotas resultantes do operador *multi-route cross*.

O operador *multi-route 3-node swap* consiste em trocar a posição de três clientes de três rotas distintas. Logo, dados três clientes $v_1 \in V$ na rota do veículo $k_1 \in R$, $v_2 \in V$ na rota do veículo $k_2 \in R$ e $v_3 \in V$ na rota do veículo $k_3 \in R$, um novo vizinho de S pode ser obtido removendo-se os arcos $(u_1,\, v_1) \in S^{k_1}$, $(v_1,\, w_1) \in S^{k_1}$, $(u_2,\, v_2) \in S^{k_2}$, $(v_2,\, w_2) \in S^{k_2}$, $(u_3,\, v_3) \in S^{k_3}$ e $(v_3,\, w_3) \in S^{k_3}$ e inserindo os arcos $(u_3,\, v_1)$ e $(v_1,\, w_3)$ em S^{k_3}, $(u_1,\, v_2)$ e $(v_2,\, w_1)$ em S^{k_1} e $(u_2,\, v_3)$ e $(v_3,\, w_2)$ em S^{k_2}. A Figura 3.8 ilustra as rotas $\langle 0, 1, 2, 8, 4, 5, 0\rangle$, $\langle 0, 6, 7, 13, 9, 10, 0\rangle$ e $\langle 0, 11, 12, 3, 14, 15, 0\rangle$ dos veículos k_1, k_2 e k_3, respectivamente, que podem ser obtidas através de uma operação de *multi-route 3-node swap* dos clientes 3, 8 e 13 nas rotas $\langle 0, 1, 2, 3, 4, 5, 0\rangle$, $\langle 0, 6, 7, 8, 9, 10, 0\rangle$ e $\langle 0, 11, 12, 13, 14, 15, 0\rangle$. Os arcos pontilhados representam os arcos eliminados das rotas.

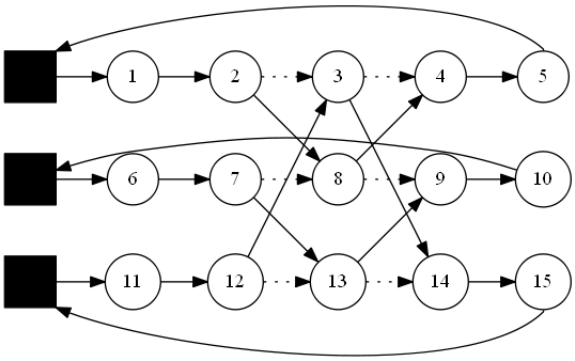

Figura 3.8. Exemplo de rotas resultantes do operador *multi-route 3-node swap*.

O operador *multi-route 3-exchange* consiste em trocar as posições de três pares de clientes consecutivos de três rotas distintas. Logo, dados três arcos $(v_1,\, d_1) \in S^{k_1}$, $(v_2,\, d_2) \in S^{k_2}$ e $(v_3,\, d_3) \in S^{k_3}$, um novo vizinho de S pode ser obtido removendo-se os arcos $(u_1,\, v_1) \in S^{k_1}$, $(d_1,\, w_1) \in S^{k_1}$, $(u_2,\, v_2) \in S^{k_2}$, $(d_2,\, w_2) \in S^{k_2}$, $(u_3,\, v_3) \in S^{k_3}$ e $(d_3,\, w_3) \in S^{k_3}$ e inserindo os arcos $(u_2,\, v_3)$ e $(d_3,\, w_2)$ em S^{k_2}, $(u_1,\, v_2)$ e $(d_2,\, w_1)$ em S^{k_1} e $(u_3,\, v_1)$ e $(d_1,\, w_3)$ em S^{k_3}. A Figura 3.9 ilustra as rotas $\langle 0, 1, 2, 8, 9, 5, 0\rangle$, $\langle 0, 6, 7, 13, 14, 10, 0\rangle$ e $\langle 0, 11, 12, 3, 4, 15, 0\rangle$ dos veículos k_1, k_2 e k_3, respectivamente, que

podem ser obtidas através de uma operação de *multi-route 3-exchange* dos arcos (3, 4), (8, 9) e (13, 14) nas rotas $\langle 0, 1, 2, 3, 4, 5, 0\rangle$, $\langle 0, 6, 7, 8, 9, 10, 0\rangle$ e $\langle 0, 11, 12, 13, 14, 15, 0\rangle$. Os arcos pontilhados representam os arcos eliminados das rotas.

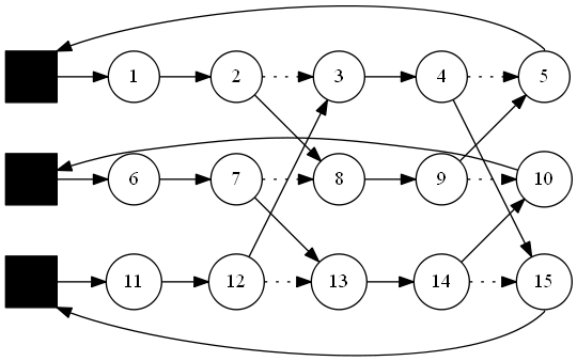

Figura 3.9. Exemplo de rotas resultantes do operador *multi-route 3-exchange*.

3.2 Heurística Construtiva

A heurística contrutiva utilizada neste trabalho é baseada na heurística PFIH [Solomon, 1987]. Ela começa com uma solução parcial onde a rota de cada veículo em R está vazia. Em cada iteração, os clientes que ainda não estão presentes na solução parcial são identificados e o custo incremental de inserção de cada um deles, em cada posição viável nesta solução é avaliado. Além disso, sempre é testado se a inserção do cliente em uma rota vazia tem custo melhor que a inserção em uma rota não vazia. Neste caso, o cliente com o menor custo incremental é inserido na sua melhor posição possível. Este procedimento para quando todos os clientes estão inseridos na solução. Como $|R| = |N|$ e os clientes só estão inseridos em posições onde nenhuma restrição é violada, as soluções fornecidas por esta heurística são viáveis para VRPMTW.

3.3 Heurística de Minimização do Número de Rotas

Assim como sugerido por Bräysy & Gendreau [2005b], VNS-VRPMTW utiliza-se de uma heurística que visa unicamente minimizar o número de rotas não vazias, chamada de *Route-Elimination*. Nesse caso, os tempos de viagem e os tempos de espera dos veículos são desconsiderados. Portanto, *Route-Elimination* utiliza uma função objetivo modificada, chamada de $f'(S)$ que contabiliza apenas o número de rotas não vazias em S. Essa heurística é baseada em VNS e é composta por um procedimento de busca

local, chamado de *Elimination-Local-Search(k, S)*, e um procedimento de perturbação, chamado de $Perturbation(k, S, \alpha)$. As rotas são ordenadas por quantidade de clientes e o procedimento *Elimination-Local-Search(k, S)* é executado na rota com o menor número de clientes. Uma vez que esse procedimento estaguina, o procedimento $Perturbation(k, S, \alpha)$ é executado. Ambos são descritos a seguir e o pseudocódigo de *Route-Elimination* é apresentado logo em seguida.

O *Elimination-Local-Search(k, S)* tem como objetivo remover todos os clientes da rota de um veículo k na solução S. Para isso, ele alterna iterativamente entre dois procedimentos: *restricted-relocate* e *random-swap*. No *restricted-relocate*, para cada cliente i na rota de k, procura-se realocar i para uma posição viável, nas rotas não vazias dos outros veículos em S, que gere o menor aumento do custo da solução. Caso não exista uma posição viável, o procedimento tenta então realocar o próximo cliente na rota. Caso ainda existam clientes na rota de k após *restricted-relocate*, é aplicado o *random-swap*. Esse procedimento visa trocar aleatoriamente os clientes na rota de k por clientes nas rotas de outros veículos, na esperança de que os novos clientes na rota de k possam ser realocados pelo *restricted-relocate* na próxima iteração do *Elimination-Local-Search(k, S)*. Cabe destacar que uma vez que um cliente é removido da rota k pelo procedimento *random-swap*, ele não pode ser inserido novamente na rota de k nas próximas iterações de *random-swap*. *Elimination-Local-Search(k, S)* termina quando não restam clientes na rota de k ou quando não é possível trocar os clientes na rota de k por outros clientes na solução.

O procedimento $Perturbation(k, S, \alpha)$ do *Route-Elimination* é aplicado para alterar aleatoriamente as rotas dos veículos em S, exceto na rota de k. O intuito é que os clientes na rota de k possam ser eventualmente realocados pelo *Elimination-Local-Search(k, S)* na solução resultante. Esse procedimento consiste em, dado um tamanho de perturbação α, selecionar aleatoriamente um dos cinco operadores de vizinhança de múltiplas rotas descritos acima e aplicá-lo η vezes a clientes selecionados aleatoriamente.

O pseudocódigo da heurística *Route-Elimination* é apresentado no Algoritmo 1. A melhor solução encontrada S^* é inicializada na linha 1, enquanto o valor do tamanho da perturbação α é atribuído a seu valor mínimo η_{min} na linha 2. O conjunto $R^S \subseteq R$ de veículos com rotas não vazias em S é inicializado na linha 3 e o veículo $k \in R^S$ cuja rota tem o menor número de clientes em S é identificado em linha 4, onde $size(k', S)$ indica o número de clientes na rota de k. Em cada iteração do ciclo nas linhas 5-19, um novo ótimo local S é gerado aplicando o procedimento *Perturbation* à solução atual S na linha 6 se existirem clientes na rota do veículo k e, em seguida, aplicando o procedimento *Elimination-Local-Search* à solução resultante S na linha 7. Se S for

melhor do que S^* (linha 8), a melhor solução encontrada e o tamanho da perturbação são atualizados nas linhas 9 e 10, respectivamente. Caso contrário, o valor de α é incrementado na linha 12 se α for menor que o seu valor máximo η_{max}. Se a rota de k está vazia ou uma condição de estagnação é atingida na linha 14, k é removido de R^S na linha 15, ou seja, a heurística não mais tenta remover clientes da rota de k. Após isso, um novo veículo cuja rota possui o menor número de clientes em R^S é selecionado na linha 16. Nesse caso, o valor de α também é redefinido para o seu menor valor η_{min} na linha 17. A condição de estagnação é alcançada após β iterações sem esvaziar a rota de k. Já a condição de parada é atendida quando θ iterações do ciclo nas linhas 5-19 são executadas ou quando o número de rotas não vazias em S^* é igual ao limite inferior T, calculado na Equação (3.1), para o número mínimo de veículos necessários para enviar todas as demandas dos clientes. Ao final deste ciclo, a melhor solução encontrada S^* é retornada na linha 20. Neste trabalho, o valor de β e θ foram definidos para 200 e 700, respectivamente, e os valores de η_{min} e η_{max} foram fixados em para 5 e 20, respectivamente. Os valores desses parâmetros foram definidos em experimentos preliminares.

$$T = \frac{1}{Q} \sum_{i \in N} q_i. \tag{3.1}$$

3.4 Heurística de Minimização da Duração de Rotas

A heurística *Route-Optimization* visa minimizar as funções objetivo (1.1) ou (1.2) sem aumentar o número de rotas não vazias. Essa heurística é baseada em VNS e é composta de um procedimento *Shaking(S, n)* e um procedimento VND(S'). Esses procedimentos são descritos a seguir. Logo em seguida, é apresentado o pseudocódigo de *Route-Optimization*.

O procedimento de perturbação do *Route-Optimization*, chamado de $Shaking(S, n)$, tem como objetivo gerar uma nova solução S' que permita ao procedimento de busca local encontrar um novo ótimo local. Ele consiste em, dado um tamanho de perturbação n, escolher aleatoriamente entre os operadores *multi-route relocate* e *multi-route swap* e aplicar o operador escolhido n vezes para clientes escolhidos aleatoriamente em S.

O procedimento $VND(S')$ consiste em um VND composto por oito buscas locais. As sete primeiras se baseiam em sete operadores de vizinhanças descritos na seção 3.1 e

Algorithm 1 *Route − Elimination*(S)

1: $S^* \leftarrow S$
2: $\eta \leftarrow \eta_{min}$
3: Seja R^S o conjunto de veículos com rotas não vazias em S.
4: $k \leftarrow argmin_{k' \in R^S} size(k', S)$
5: **while** Condição de parada não atingida **do**
6: **if** $size(k, S) > 0$ **then** $S \leftarrow Perturbation(k, S, \alpha)$
7: $S \leftarrow$ *Elimination-Local-Search(k, S)*
8: **if** $f'(S) < f'(S^*)$ **then**
9: $S^* \leftarrow S$
10: $\eta \leftarrow \eta_{min}$
11: **else if** $\eta < \eta_{max}$ **then**
12: $\eta \leftarrow \alpha + 1$
13: **end if**
14: **if** Se $size(k, S) = 0$ **or** condição de estagnação é atingida **then**
15: $R^S \leftarrow R^S \setminus \{k\}$
16: $k \leftarrow argmin_{k' \in R^S} size(k', S)$
17: $\eta \leftarrow \eta_{min}$
18: **end if**
19: **end while**
20: **return** S^*

a oitava consiste no procedimento *restricted-relocate* descrito na seção 3.3. As sete buscas locais são executadas na seguinte ordem: (i) *single-route relocate*, (ii) *single-route 2-exchange*, (iii) *multi-route relocate*, (iv) *multi-route swap*, (v) *multi-route cross*, (vi) *multi-route 3-node swap* e (vii) *multi-route 3-exchange*. Nessas buscas locais, apenas são executadas as operações cuja duração das rotas é pelo menos tão curta quanto na solução corrente. Isso é, se um movimento gera um vizinho cujo tempo de viagem de alguma rota é pior que o tempo de viagem da solução atual, ele é descartado sem que o tempo de espera seja computado. Isso é feito para minimizar o custo computacional da avaliação dos vizinhos. Esse procedimento se distingue do VND clássico proposto na literatura em [Mladenović & Hansen, 1997], pois procedimento só passa para próxima busca local quando encontra o ótimo local na busca local corrente. Porém, caso uma nova solução melhor que a solução corrente seja encontrada, ao fim da última busca local, o procedimento se repete. Esse procedimento só finaliza quando um ótimo local comum a todas buscas locais seja encontrado. A última busca local do VND consiste no procedimento *restricted-relocate*, pois a heurística *Route-Elimination* não garante que o número mínimo de rotas para uma instância seja atingido. Portanto, se o número de rotas na solução atual for maior que o limite inferior T, calculado em (3.1), esse procedimento é executado ao fim das demais buscas locais.

O pseudocódigo da heurística *Route-Optimization* é apresentado no Algoritmo 2. A melhor solução encontrada S^* é inicializada na linha 1, enquanto o valor do tamanho da perturbação n é atribuído ao seu valor mínimo η_{min} na linha 2. Em cada iteração do ciclo nas linhas 3-19, um novo ótimo local S' é gerado aplicando o procedimento *Shaking* à solução atual S na linha 4 e, em seguida, aplicando o procedimento VND à solução resultante S' na linha 5. Caso a função objetivo utilizada seja a descrita em (1.1), o procedimento MBTSA, proposto por Belhaiza et al. [2014] e descrito no Capítulo 2, é aplicado à S', na linha 6, para minimizar o tempo de espera. Caso contrário, ele não é executado, pois a função objetivo (1.2) não considera o tempo de espera. Se S' for melhor do que S (linha 7), a solução atual e a melhor solução encontrada, bem como o tamanho da perturbação, são atualizadas nas linhas 8 a 10, respectivamente. Caso contrário, o valor de n é incrementado na linha 12 se n for menor que o seu valor máximo η_{max}. Se uma condição de estagnação é atingida na linha 14, a solução corrente é substituída, na linha 16, por uma solução escolhida aleatoriamente, na linha 15, dentre as γ melhores soluções encontradas até o momento. Nesse caso, o valor de n também é redefinido para o seu menor valor η_{min}. A condição de estagnação é atingida após λ iterações sem melhorar a melhor solução encontrada. Já a condição de parada é definida como δ iterações do ciclo nas linhas 3-19 sem melhora da melhor solução conhecida S^*. Ao final deste ciclo, a melhor solução encontrada S^* é retornada na linha 20. Neste trabalho, os valores de γ, λ e δ foram definidos para 10, 60 e 500, respectivamente, e os valores de η_{min} e η_{max} foram fixados em para 5 e 20, respectivamente.

Algorithm 2 Route-Optimization(S)

1: $S^* \leftarrow S$
2: $n \leftarrow \eta_{min}$
3: **while** Condição de parada não é atingida **do**
4: $S' = Shaking(S, \text{n})$
5: $S' \leftarrow VND(S')$
6: **if** Se a função objetivo é a (1.1) **then** $S' \leftarrow MBTSA(S')$
7: **if** $f(S') < f(S)$ **then**
8: $S \leftarrow S'$
9: $S^* \leftarrow best(S, S^*)$
10: $n \leftarrow \eta_{min}$
11: **else if** $n < \eta_{max}$ **then**
12: $n \leftarrow n + 1$
13: **end if**
14: **if** Condição de estagnação é atingida **then**
15: Seja Γ o conjunto das melhores γ soluções encontradas até o momento.
16: Atribua randomicamente a S uma das soluções de Γ.
17: $n \leftarrow \eta_{min}$
18: **end if**
19: **end while**
20: **return** S^*

Capítulo 4

Experimentos Computacionais

Esta seção compara os resultados da heurística VNS-VRPMTW, que apenas explora regiões viáveis do espaço de busca, com os resultados da heurística HVNTS [Belhaiza et al., 2014], que lida prioritariamente com soluções inviáveis. A primeira foi implementada em C++ e compilada com o GNU *gcc* versão 6.3. A heurística HVNTS também foi implementada em C++ e os resultados apresentados nesta seção foram obtidos pelo executável do código fornecido pelos autores de [Belhaiza et al., 2014]. O HVNTS é limitado a 10000 iterações, como proposto em [Belhaiza et al., 2014]. Experimentos computacionais foram realizados em um uma máquina Intel i5-3230M com $2.60GHz$ de velocidade de clock e $8GB$ de RAM.

Neste trabalho, são utilizadas as 72 instâncias propostas em [Belhaiza et al., 2014]. As instâncias, todas de 100 clientes, são baseadas nas instâncias propostas por Solomon [1987] e se subdividem em três tipos: C, R e RC. As instâncias do tipo C possuem clientes dispostos de forma a gerar grupos de clientes geograficamente próximos uns dos outros, porém os grupos estão geograficamente distantes uns dos outros. Já nas instâncias do tipo R, os clientes estão dispostos aleatoriamente no espaço. Nas instâncias do tipo RC, parte dos clientes formam grupos de clientes geograficamente próximos uns dos outros e o restante dos clientes estão dispostos aleatoriamente no espaço. Além disso, cada tipo se subdivide em duas classes. A diferença das instâncias da classe 2 para as instâncias da classe 1 é um valor maior para o limite máximo de duração de rota D e um maior limite de capacidade dos veículos Q.

Além dessas características, as instâncias propostas por Belhaiza et al. [2014] se dividem em dois grupos M e P. No grupo M, cada cliente possui janelas de tempo não-sobrepostas, isso é, há um intervalo de tempo entre cada janela no qual o cliente não pode ser atendido. No segundo, alguns clientes podem possuir janelas de tempo que se sobrepõem. Dessa forma, o primeiro grupo é composto pelos conjuntos de instâncias

cm1, rm1, rcm1, cm2, rm2 e rcm2, com cada conjunto possuindo oito instâncias. Já o grupo P é composto pelos conjuntos de instâncias pc1, pr1, prc1, pc2, pr2 e prc2, com cada conjunto possuindo quatro instâncias.

Belhaiza et al. [2014] também considera duas variantes de VRPMTW. Na primeira, ele considera a função objetivo (1.1) que minimiza também o tempo de espera em cada cliente. Na segunda, ele considera a função objetivo (1.2) que não minimiza o tempo de espera em cada cliente. Primeiramente, são apresentados os experimentos realizados para a variante que utiliza a função objetivo (1.1). Em seguida, são apresentados os resultados para a variante que utiliza a função objetivo (1.2).

4.1 Resultados para a variante cuja função objetivo é (1.1)

O primeiro experimento realizado avalia o impacto das buscas locais baseadas nos operadores de vizinhança de múltiplas rotas no desempenho do procedimento VND executado na linha 5 do algoritmo 2 na variante de VNS-VRPMTW que utiliza a função objetivo (1.1). Experimentos preliminares demonstraram que as buscas locais de rota única são fundamentais no desempenho do VND. Além disso, essas buscas locais possuem baixo custo computacional. Sendo assim, o impacto da remoção das buscas locais que utilizam operadores de rota única não estão inclusos neste experimento. Portanto, sete variações de VNS-VRPMTW são estudadas. VNS-ALL utiliza todas as sete buscas locais baseadas nos sete operadores de vizinhanças definidos na seção 3.1 no VND. VNS-R é igual ao VNS-ALL, mas não utiliza a busca local baseada no operador *multi-route relocate* no procedimento VND. VNS-S, por sua vez, é uma variação de VNS-ALL que não utiliza a busca local baseada no operador *multi-route swap* no procedimento de VND. VNS-C é uma variação de VNS-ALL que não utiliza a busca local baseada no operador *multi-route cross* no procedimento de VND. VNS-3N é uma variação de VNS-ALL que não utiliza a busca local baseada no operador *multi-route 3-node swap*. Já VNS-3E é igual ao VNS-ALL, mas não utiliza a busca local baseada no operador *multi-route 3-exchange*. Por fim, VNS-3NE é a variação de VNS-ALL que não utiliza as buscas locais baseadas nos operadores *multi-route 3-node swap* e *multi-route 3-exchange*. As heurísticas foram executadas dez vezes para todos os grupos de instâncias.

Os resultados para esse experimento são apresentados na Tabela 4.1. Os grupos de instâncias são apresentadas na primeira coluna. Na segunda e terceira colunas, são apresentadas as médias dos custos das soluções encontradas e dos tempos de execução

de VNS-ALL respectivamente. A quarta coluna mostra o desvio relativo percentual $dev(f) = (f^{ALL} - f^R)/f^{ALL}$ entre a média de custo das soluções obtidas por VNS-R (f^R) e a média de custo das soluções obtidas por VNS-ALL (f^{ALL}). Vale salientar que valores positivos para esta métrica indicam que as soluções de VNS-R são melhores que as soluções de VNS-ALL. A quinta coluna mostra o desvio relativo percentual $dev(t) = (t^{ALL} - t^R)/t^{ALL}$ entre a média de tempo de execução de VNS-R (t^R) e a média de tempo de execução de VNS-ALL (t^{ALL}). Vale salientar que valores positivos para esta métrica indicam que os tempos de execução de VNS-R são menores que os tempos de execução de VNS-ALL em média. Os mesmos valores são apresentados nas sexta e sétima colunas, respectivamente, para VNS-S, nas oitava e nona colunas, respectivamente, para VNS-C, nas décima e décima primeira colunas, respectivamente, para VNS-3N, nas décima segunda e décima terceira colunas, respectivamente, para VNS-3E e, finalmente, nas duas últimas colunas para VNS-3NE.

Pode ser observado que removendo a busca local que utiliza o operador *multi-route 3-exchange*, as soluções são, em média, $0,07\%$ melhores e ao mesmo tempo, possui tempo de execução $21,61\%$ menor que VNS-ALL. Já a remoção das demais buscas locais levam a uma piora na qualidade das soluções por exceção de VNS-S. Porém, essa variante não foi escolhida porque a redução no tempo de excução não foi tão expressiva quanto a redução em VNS-3E. A variante VNS-3NE é a que leva à maior redução do tempo de execução, $40,31\%$, porém as soluções encontradas são aproximadamente $0,92\%$ piores do que as encontradas por VNS-ALL. Portanto, para verificar dentre as variantes VNS-ALL, VNS-3E e VNS-3NE qual é a de melhor desempenho, foi utilizada a técnica de verificação gráfica TTTPlot. O teste consistiu de duzentas execuções das variantes para um conjunto de instâncias e cada execução foi limitada a até seiscentos segundos de execução sem atingir o alvo.

Os resultados desse experimento para as instâncias rm101, cm101, pr101 e pc101 são apresentados nos gráficos nas Figuras 4.1, 4.2, 4.3 e 4.4. Pode ser observado que a variante VNS-3E possui melhor desempenho em todos os casos testados exceto para a instância pr101. Portanto, a versão de VNS-VRPMTW utilizada nos experimentos a seguir para a variante que utiliza a função objetivo (1.1) é a VNS-3E que não inclui a busca local que utiliza o operador *multi-route 3-exchange* no procedimento VND.

O segundo experimento realizado verifica o desempenho do procedimento VND executado na linha 5 do algoritmo 2 proposto nessa dissertação para a variante da heurística proposta que utiliza a função objetivo (1.1). Para tanto, três variações de VNS-VRPMTW são estudadas. VNS-VNDP utiliza o procedimento VND proposto nessa dissertação. A segunda variação, chamada de VNS-VNDL, utiliza na busca local do algoritmo 2 o procedimento de VND clássico proposto em [Mladenović & Hansen,

Tabela 4.1. Resultados médios de 10 execuções independentes das heurísticas VNS-ALL, VNS-R, VNS-S, VNS-C, VNS-3N, VNS-3E e VNS-3NE com o uso da função objetivo (1.1)

Grupos	VNS-ALL		VNS-R		VNS-S		VNS-C		VNS-3N		VNS-3E		VNS-3NE	
	f(S)	Tempo (s)	dev(f) %	dev(t) %	dev(f) %	dev(t) %	dev(f) %	dev(t) %	dev(f) %	dev(t) %	dev(f) %	dev(t) %	dev(f) %	dev(t) %
rm1	2785,13	481,90	-5,41	-7,79	-0,02	-26,49	-0,83	5,01	-0,82	31,17	-0,13	11,80	-0,68	34,68
rm2	2865,25	409,46	-0,28	-10,73	0,86	0,43	-0,27	4,10	0,11	4,50	0,04	-41,05	0,03	6,12
cm1	3562,75	289,38	-2,38	22,54	-0,02	19,35	-0,49	4,89	-2,83	41,41	-0,18	41,87	-3,85	83,00
cm2	4326,00	309,01	-2,44	4,20	-0,02	1,05	-0,84	-15,60	-0,75	-2,92	-0,13	20,64	-0,95	31,50
rcm1	3338,50	388,74	-3,78	13,41	0,28	8,29	-0,39	6,75	-0,18	31,45	-0,07	40,59	-0,21	61,87
rcm2	3618,63	467,73	-7,69	-10,80	1,52	-13,46	-0,29	8,52	0,25	19,09	0,63	10,49	-0,41	29,32
pr1	2674,75	811,56	-1,25	27,05	-0,01	-8,71	-0,24	5,79	-0,17	55,98	-0,09	64,35	-0,17	44,70
pr2	2707,00	350,81	-2,86	-0,81	-1,12	22,06	-1,32	-0,10	-0,95	-24,82	0,10	-38,60	-0,35	20,26
pc1	2864,25	464,02	-6,29	43,27	-0,27	52,68	-1,41	-25,27	-2,48	33,89	-0,18	75,59	-3,28	95,84
pc2	2738,75	490,23	-3,66	-90,32	0,05	10,44	-1,21	-54,76	-0,16	-31,10	0,19	25,25	-1,01	8,13
prc1	2935,00	186,40	-3,41	-51,07	1,18	4,77	-0,76	-25,73	-0,55	24,92	0,79	41,45	-0,82	54,50
prc2	2701,25	272,49	-17,01	-39,86	-0,14	-7,60	-0,23	-27,59	0,10	-16,35	-0,11	6,87	0,09	18,46
		Média	-4,71	-8,41	0,19	5,23	-0,69	-9,50	-0,70	13,93	0,07	21,61	-0,91	46,24

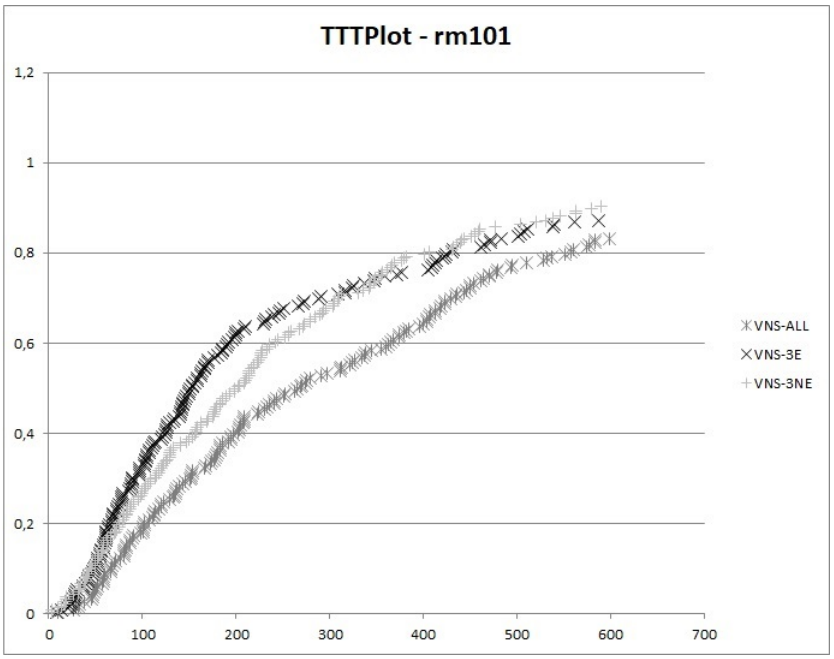

Figura 4.1. TTTPlot de 200 execuções das variantes VNS-ALL, VNS-3E e VNS-3NE para a instância rm101.

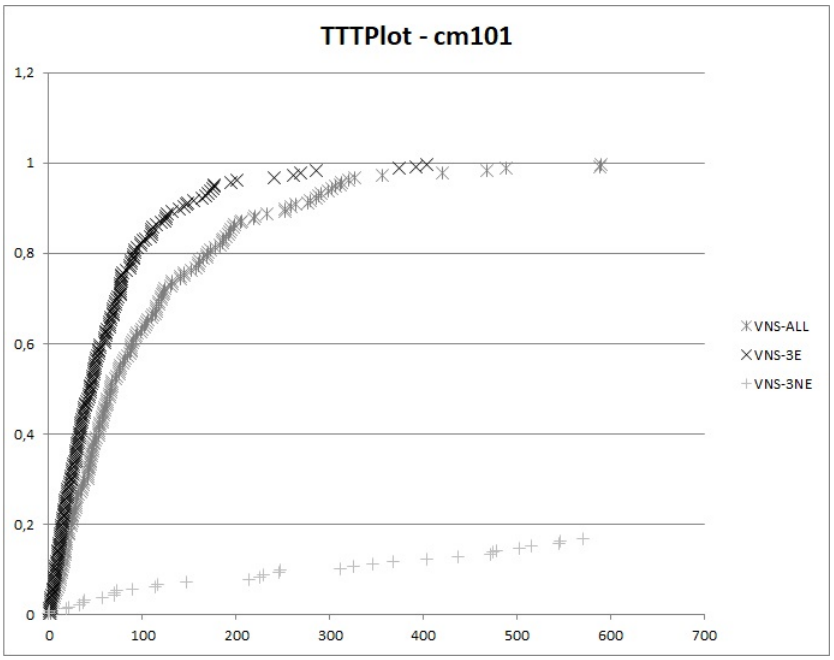

Figura 4.2. TTTPlot de 200 execuções das variantes VNS-ALL, VNS-3E e VNS-3NE para a instância cm101.

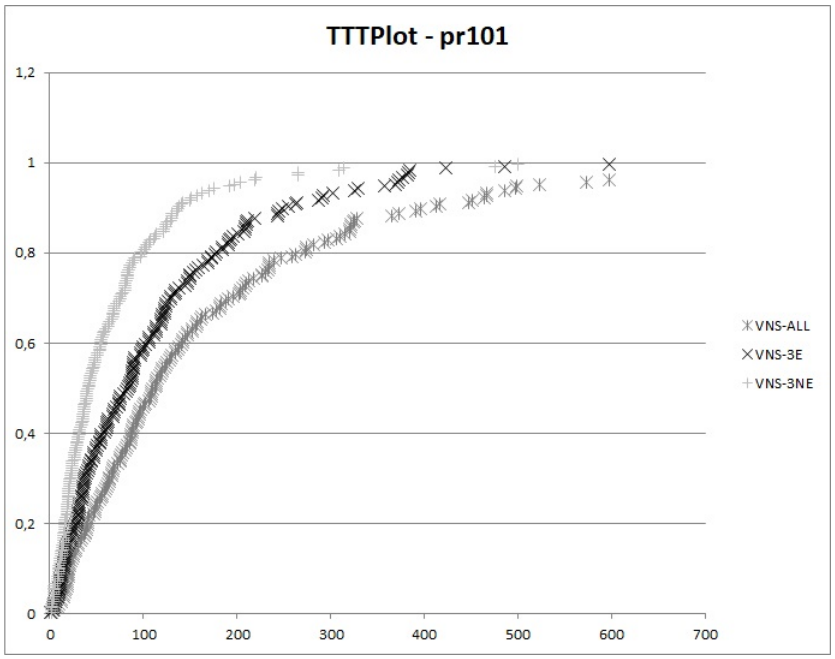

Figura 4.3. TTTPlot de 200 execuções das variantes VNS-ALL, VNS-3E e VNS-3NE para a instância pr101.

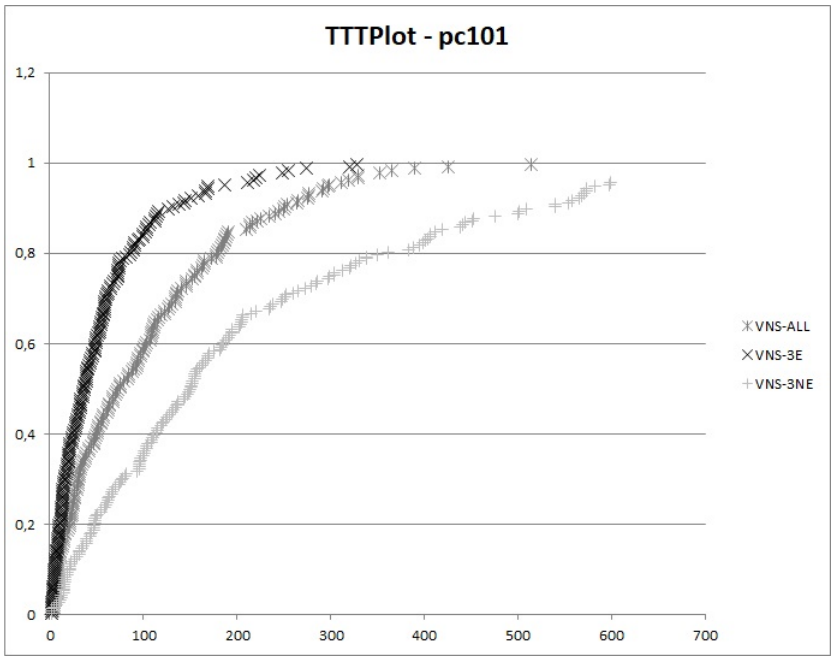

Figura 4.4. TTTPlot de 200 execuções das variantes VNS-ALL, VNS-3E e VNS-3NE para a instância pc101.

Tabela 4.2. Resultados médios de 10 execuções independentes das heurísticas VNS-VNDP, VNS-VNDL e VNS-VNDR com o uso da função objetivo (1.1)

Grupos	VNS - VNDP			VNS - VNDL			VNS - VNDR		
	f(S)	Tempo (s)	Tot. Iter	dev(f)%	dev(t)%	dev(i)%	dev(f)%	dev(t)%	dev(i)%
rm1	2789,13	389,26	937,25	-1,68	11,51	-9,94	-1,98	29,77	-9,02
rm2	2863,00	564,61	946,38	-1,91	32,83	10,45	-1,19	18,32	9,29
cm1	3568,88	175,10	913,75	-0,61	15,11	8,86	-2,64	39,85	4,84
cm2	4331,63	240,71	818,00	-2,37	16,64	-0,91	-2,70	-12,91	-3,63
rcm1	3340,75	222,04	910,25	-1,11	-2,89	-11,62	-1,30	-26,32	-11,65
rcm2	3603,63	386,90	834,25	-8,51	11,48	-4,22	-7,01	12,19	-6,47
pr1	2677,25	289,17	885,25	-2,30	8,08	6,51	-2,39	-36,99	0,15
pr2	2704,25	455,93	993,75	-1,53	13,53	6,63	-1,75	1,07	3,25
pc1	2869,25	98,15	839,50	-5,82	15,98	-3,48	-5,99	28,13	-7,99
pc2	2733,50	366,27	928,25	-6,49	3,98	-0,24	-10,02	11,19	8,47
prc1	2911,75	105,03	846,50	-5,24	-11,00	-1,38	-8,77	-49,73	5,79
prc2	2704,25	246,69	925,50	-4,24	-6,54	1,19	-4,62	-45,66	-1,70
			Média:	-3,48	9,06	0,15	-4,20	-2,59	-0,72

1997]. A terceira variação, chamada de VNS-VNDR, consiste de um procedimento de VND em que, toda vez que a solução corrente é melhorada, uma das buscas locais é escolhida aleatoriamente para dar continuidade a otimização. As heurísticas foram executadas dez vezes para todos os grupos de instâncias.

Os resultados para esse experimento são apresentados na Tabela 4.2. Os grupos de instâncias são apresentadas na primeira coluna. Na segunda, terceira e quarta colunas, são apresentadas as médias dos custos das soluções encontradas, dos tempos de execução e do número total de iterações de VNS-VNDP respectivamente. A quinta coluna mostra o desvio relativo percentual $dev(f) = (f^{VNDP} - f^{VNDL})/f^{VNDP}$ entre a média de custo das soluções obtidas por VNS-VNDL (f^{VNDL}) e a média de custo das soluções obtidas por VNS-VNDP (f^{VNDP}). Vale salientar que valores positivos para esta métrica indicam que as soluções de VNS-VNDP são melhores que as soluções de VNS-VNDL. A sexta coluna mostra o desvio relativo percentual $dev(t) = (t^{VNDP} - t^{VNDL})/t^{VNDP}$ entre a média de tempo de execução de VNS-VNDL (t^{VNDL}) e de VNS-VNDP (t^{VNDP}). Vale salientar que valores positivos para esta métrica indicam que os tempos de execução de VNS-VNDP são menores que os tempos de execução de VNS-VNDL. A sétima coluna mostra o desvio relativo percentual $dev(i) = (i^{VNDP} - i^{VNDL})/i^{VNDP}$ entre a média do número de iterações de VNS-VNDL (i^{VNDL}) e o número de iterações de VNS-VNDP (i^{VNDP}). Vale salientar que valores positivos para esta métrica indicam que VNS-VNDP realiza menos iterações que VNS-VNDL. Os mesmos valores são apresentados na oitava, nona e décima colunas para VNS-VNDR.

Pode ser observado que VNS-VNDP encontra soluções $3,48\%$ e $4,20\%$ melhores

em média que VNS-VNDL e VNS-VNDR respectivamente. Isso se dá porque VNS-VNDP só sai de uma busca local quando encontra o ótimo local dessa. Isso permite que a heurística escape do ótimo local das outras buscas locais com mais facilidade do que as outras variações. Por outro lado, o tempo de execução aumenta em $9,06\%$ em média em relação a VNS-VNDL. Isso se dá porque, em VNS-VNDP, só há a troca de buscas locais quando se encontra o ótimo local da busca local corrente, o que faz com que a heurística passe mais tempo em nas buscas locais de maior tamanho. Em comparação com VNS-VNDR, o tempo de execução de VNS-VNDP é $2,59\%$ melhor. Isso se dá porque VNS-VNDR escolhe a próxima busca local aleatoriamente e, portanto, buscas locais cuja vizinhança é maior podem ser escolhidas primeiro que aquelas de menor tamanho, o que leva a uma maior demora para achar um ótimo local comum a todas as buscas locais. Isso também explica porque VNS-VNDR é mais rápido em sete dos doze grupos, pois, nesses casos, o procedimento pode ter escolhido as buscas locais de menor tamanho primeiramente. A variação do número de iterações entre as variantes de VNS-VRPMTW não é muito significativa. Portanto, a variante de VNS-VRPMTW utilizada foi VNS-VNDP.

O terceiro experimento realizado procura definir o melhor critério de parada para VNS-VRPMTW quando utiliza-se a função objetivo (1.1). Para tanto, foram propostas três variantes de VNS-VRPMTW. As variantes, VNS-100, VNS-300 e VNS-500, têm como critério de parada do *loop* na linha 3 do algoritmo 2 100, 300 e 500 iterações sem melhora da melhor solução conhecida respectivamente. As heurísticas foram executadas dez vezes para todos os grupos de instâncias e comparadas com os resultados de 10 execuções de HVNTS conforme proposto em [Belhaiza et al., 2014].

Os resultados desse experimento são apresentados na Tabela 4.3. Na primeira coluna, são apresentados os grupos de instâncias usados. Na segunda e terceira, são apresentadas as médias dos custos das melhores soluções encontradas e dos tempos de execução para dez execuções independentes de HVNTS. Na quarta coluna é apresentado o desvio relativo percentual $dev(f) = (f^{HVNTS} - f^{100})/f^{HVNTS}$ entre a média de custo das soluções obtidas por VNS-100 (f^{100}) e a média de custo das soluções obtidas por HVNTS (f^{HVNTS}). Vale salientar que valores positivos para esta métrica indicam que as soluções de VNS-100 são melhores que as soluções de HVNTS. A quinta coluna mostra o desvio relativo percentual $dev(t) = (t^{HVNTS} - t^{100})/t^{HVNTS}$ entre a média de tempo de execução de VNS-100 (t^{100}) e a média de tempo de execução de HVNTS (t^{HVNTS}). Nessa métrica, valores positivos indicam que o tempo de execução de VNS-VRPMTW é menor que o tempo de execução de HVNTS. Os mesmos valores são apresentados nas sexta e sétima colunas para VNS-300 e nas oitava e nona colunas para VNS-500.

Tabela 4.3. Resultados médios de 10 execuções independentes das heurísticas VNS-100, VNS-300 e VNS-500 com o uso da função objetivo (1.1) em comparação com os resultados de HVNTS

Grupos	HVNTS		VNS-100		VNS-300		VNS-500	
	f(S)	Tempo (s)	dev(f) %	dev(t) %	dev(f) %	dev(t) %	dev(f) %	dev(t) %
rm1	2831,13	108,96	-0,27	-13,05	0,98	-300,16	1,45	-273,94
rm2	3525,13	107,95	15,35	-62,89	15,78	-415,40	17,81	-447,94
cm1	3453,29	159,36	-3,73	69,42	-2,22	-45,60	-2,14	0,97
cm2	5098,50	147,60	14,38	42,26	14,98	-111,80	15,03	-62,73
rcm1	3468,75	94,89	1,83	10,80	3,21	-148,15	3,64	-137,94
rcm2	3793,38	140,04	-0,15	27,78	2,38	-176,38	5,07	-181,30
pr1	2695,75	111,46	0,32	15,43	0,66	-174,97	0,68	-160,71
pr2	2743,50	133,24	-1,15	45,27	0,14	-28,46	1,44	-255,08
pc1	2867,50	143,49	-2,16	83,60	-0,66	3,14	-0,06	31,06
pc2	2751,50	157,00	-1,40	25,44	-0,41	-223,41	0,62	-134,70
prc1	2811,00	101,85	-6,14	57,55	-4,45	-37,80	-3,59	-3,24
prc2	2772,00	181,55	0,19	47,75	2,51	-63,74	2,42	-45,31
		Média:	1,42	29,11	2,74	-143,56	3,53	-139,24

Pode ser observado que, em todos os casos, as variantes de VNS-VRPMTW encontram, em média, soluções melhores que o HVNTS. Porém, deve-se observar que, na variante VNS-100, em apenas cinco dos doze grupos de instância, a variante VNS-100 encontra soluções melhores que as encontradas por HVNTS. A média de $1,42\%$ nesse caso é devido aos grupos de instãncia rm2 e cm2, pois a variante é capaz de eliminar um veículo a mais que HVNTS. Nas demais variantes, VNS-VRPMTW já encontra soluções na maioria dos grupos. Portanto, a variante escolhida para ser usada no restante dos testes foi VNS-500, pois ela encontra melhores soluções na maioria dos casos e porque o tempo de execução gasto não varia muito em relação a VNS-300.

O quarto experimento realizado observa o impacto do algoritmo MBTSA no despenho de VNS-VRPMTW. Para tanto, foram propostas duas variantes da heurística VNS-VRPMTW: (i) VNS+MBTSA que executa MBTSA na linha 6 do algoritmo 2 e (ii) VNS−MBTSA que não executa o MBTSA. Cada variante foi executada 10 vezes sobre todos os grupos de instâncias.

Os resultados deste experimento são apresentados na Tabela 4.4. Na primeira coluna, são apresentados os grupos de instâncias utilizados. Da segunda a quarta colunas, são apresentadas, respectivamente, as médias dos custos das soluções encontradas, dos tempos de execução e dos números de iterações realizadas por VNS+MBTSA. Na quinta coluna, é apresentado o desvio relativo percentual $dev(f) = (f^+ - f^-)/f^+$ entre a média de custo das soluções obtidas por VNS−MBTSA (f^-) e a média de custo das soluções obtidas por VNS+MBTSA (f^+). Vale salientar que valores positivos para esta métrica indicam que as soluções de VNS−MBTSA são melhores que

Tabela 4.4. Resultados médios de 10 execuções independentes das heurísticas VNS+MBTSA e VNS-MBTSA

Grupos	VNS+MBTSA			VNS−MBTSA		
	f(S)	Tempo(s)	Tot. Iter.	dev(f) %	dev(t) %	dev(i) %
rm1	2789,13	389,26	937,25	-1,47	17,09	-5,35
rm2	2863,00	564,61	946,38	-4,77	16,15	10,15
cm1	3568,88	175,10	913,75	-10,43	-30,96	2,59
cm2	4331,63	240,71	818,00	-6,17	6,90	-2,27
rcm1	3340,75	222,04	910,25	-1,79	10,64	1,23
rcm2	3603,63	386,90	834,25	-6,13	14,49	-1,52
pr1	2677,25	289,17	885,25	-1,01	-1,80	7,45
pr2	2704,25	455,93	993,75	-0,42	35,40	26,76
pc1	2869,25	98,15	839,50	-2,85	25,62	58,22
pc2	2733,50	366,27	928,25	-2,22	-11,65	-17,56
prc1	2911,75	105,03	846,50	-0,24	-7,74	-12,57
prc2	2704,25	246,69	925,50	-0,76	-6,13	18,24
			Média:	-3,19	5,67	7,11

as soluções de VNS+MBTSA. A sexta coluna mostra o desvio relativo percentual $dev(t) = (t^+ - t^-)/t^+$ entre a média de tempo de execução de VNS−MBTSA (t^-) e a média de tempo de execução de VNS+MBTSA (t^+). Nessa métrica, valores positivos indicam que o tempo de execução de VNS−MBTSA é menor que o tempo de execução de VNS+MBTSA. Na sétima coluna, é apresentado o desvio relativo percentual $dev(i) = (i^+ - i^-)/i^+$ entre a média de número de iterações de VNS−MBTSA (i^-) e a média de número de iterações de VNS+MBTSA (i^+). Vale salientar que valores positivos para esta métrica indicam que o número de iterações de VNS−MBTSA são menores que o número de iterações de VNS+MBTSA em média.

Pode-se observar que VNS−MBTSA encontra soluções, em média, $3,19\%$ piores do que as encontradas pelo VNS+MBTSA. Em relação ao tempo de execução, o VNS−MBTSA foi, em média, $5,67\%$ mais rápido que o VNS+MBTSA. Isso se dá pois a complexidade do MBTSA é um produtório e, mesmo executando uma vez por iteração, já acaba sendo custoso para a heurística. Já em relação ao número de iterações, o VNS−MBTSA realizou $7,11\%$ iterações a menos que o VNS+MBTSA. Portanto, MBTSA se mostra necessário para que a heurística VNS-VRPMTW encontre melhores soluções e esse algoritmo é usado na heurística proposta nos demais experimentos.

O quinto experimento verifica se a estratégia de se avaliar somente o custo de vizinhos que aprimorem o tempo de viagem nas buscas locais do VND leva a um melhor desempenho da heurística proposta. Para tanto, três versões do VNS-VRPMTW foram avaliadas. A primeira, chamada de VNS-0%, é a versão do VNS-VRPMTW em que as

Tabela 4.5. Resultados médios de 10 execuções independentes das heurísticas VNS-0%, VNS-5% e VNS-10%

Grupos	VNS-0%			VNS-5%			VNS-10%		
	f(S)	Tempo (s)	Tot. Iter.	dev(f) %	dev(t) %	dev(i) %	dev(f) %	dev(t) %	dev(i) %
rm1	2789,13	389,26	937,25	-1,90	9,15	-1,86	-2,42	7,88	-0,96
rm2	2863,00	564,61	946,38	-1,55	-7,55	10,96	-1,52	-12,42	12,79
cm1	3568,88	175,10	913,75	-0,25	-19,20	-2,55	-0,60	-44,11	-6,33
cm2	4331,63	240,71	818,00	-1,40	-12,98	1,01	-1,53	-79,40	-5,46
rcm1	3340,75	222,04	910,25	-1,62	3,81	-8,39	-1,22	-6,34	-18,81
rcm2	3603,63	386,90	834,25	-1,69	-17,44	-3,16	-5,48	-44,06	7,14
pr1	2677,25	289,17	885,25	-1,05	-10,25	1,26	-0,84	-64,10	-4,58
pr2	2704,25	455,93	993,75	-0,37	30,93	11,40	-0,53	14,94	0,72
pc1	2869,25	98,15	839,50	-3,54	-28,97	7,55	-3,91	-147,51	13,87
pc2	2733,50	366,27	928,25	-2,44	-60,80	5,77	-1,82	-122,59	-3,00
prc1	2911,75	105,03	846,50	0,68	-65,58	-8,49	0,27	-98,00	-0,93
prc2	2704,25	246,69	925,50	-0,46	-14,79	1,69	-1,31	-42,96	8,43
			Média:	-1,30	-16,14	1,27	-1,74	-53,22	0,24

buscas locais avaliam somente os custos de vizinhos cujo tempo de viagem é menor ou igual àquele da solução corrente. A segunda, chamada de VNS-5%, consiste em uma versão do VNS-VRPMTW em que as buscas locais avaliam os custos de vizinhos que possam piorar em até 5% o tempo de viagem da solução corrente. Entretanto, vale salientar que o vizinho só é aceito se o custo total da solução é minimizado. A terceira, chamada de VNS-10%, consiste em uma versão do VNS-VRPMTW semelhante a VNS-5%, porém a tolerância em relação à piora do tempo de viagem é de 10%. Cada variante foi executada 10 vezes sobre os grupos de instâncias pr1, pr2, pc1, pc2, prc1 e prc2.

Os resultados desse experimento são apresentados na Tabela 4.5. A primeira coluna apresenta os grupos de instâncias utilizados. Da segunda a quarta colunas, são apresentadas as médias dos custos das soluções, dos tempos de execução e dos números de iterações feitas por VNS-0%. Na quinta coluna, é apresentado o desvio relativo percentual $dev(f) = (f^0 - f^5)/f^0$ entre a média de custo das soluções obtidas por VNS-5% (f^5) e a média de custo das soluções obtidas por VNS-0% (f^0). Vale salientar que valores positivos para esta métrica indicam que as soluções de VNS-5% são melhores que as soluções de VNS-0%. A sexta coluna mostra o desvio relativo percentual $dev(t) = (t^0 - t^5)/t^0$ entre a média de tempo de execução de VNS-5% (t^5) e a média de tempo de execução de VNS-0% (t^0). Nessa métrica, valores positivos indicam que o tempo de execução de VNS-5% é menor que o tempo de execução de VNS-0%. Na sétima coluna, é apresentado o desvio relativo percentual $dev(i) = (i^0 - i^5)/i^0$ entre a média do número de iterações de VNS-5% (i^5) e a média do número de iterações de VNS-0% (i^0). Vale salientar que valores positivos para esta métrica indicam que o número de iterações de VNS-5% são menores que o número de iterações de VNS-0% em média. Os mesmos valores são apresentados na oitava, nona e décima colunas para

VNS-10%.

Pode-se observar que VNS-5% encontra, em média, soluções de custo $1,30\%$ piores que as soluções encontradas pelo VNS-0%. Também houve um aumento médio de $16,14\%$ no tempo de execução e uma redução de $1,27\%$ no número de iterações. O VNS-10% encontra, em média, soluções de custo ainda piores $(1,74\%)$ que o custo das soluções encontradas pelo VNS-0%. Além disso, houve um aumento de $53,22\%$ no tempo de execução e uma redução de $0,24\%$ no número de iterações. A piora na qualidade das soluções se dá porque as versões VNS-5% e VNS-10% exploram mais vizinhos que não não aprimorantes em relação ao tempo de viagem de soluções, o que também implica no aumento do tempo de execução. Já em VNS-0%, só movimentos que levam a um melhor tempo de viagem são explorados e a otimização do tempo de espera é feita apenas pelo MBTSA, o que permite uma exploração mais rápida e eficaz do espaço de busca. Pode-se concluir que quanto maior o aumento da tolerância, mais movimentos são avaliados nas buscas locais, o que leva a um maior consumo de tempo por parte da heurística. Portanto, a versão utilizada no VNS-VRPMTW foi o VNS-0%.

O sexto experimento compara os resultados de VNS-VRPMTW com aqueles de HVNTS para a versão de VRPMTW cuja função objetivo é (1.1). As duas heurísticas foram executadas 10 vezes sobre todos os grupos de instâncias. Os resultados desse experimento são apresentados na Tabela 4.6. Na primeira coluna, são apresentados os grupos de instâncias usados. Na segunda e terceira, são apresentadas as médias dos custos das melhores soluções encontradas e dos tempos de execução para dez execuções independentes de HVNTS. Na quarta e quinta, são apresentadas as médias dos custos das melhores soluções encontradas e dos tempos de execução para dez execuções independentes de VNS-VRPMTW. Na sexta coluna é apresentado o desvio relativo percentual $dev(f) = (f^{HVNTS} - f^{VNS})/f^{HVNTS}$ entre a média de custo das soluções obtidas por VNS-VRPMTW (f^{VNS}) e a média de custo das soluções obtidas por HVNTS (f^{HVNTS}). Vale salientar que valores positivos para esta métrica indicam que as soluções de VNS-VRPMTW são melhores que as soluções de HVNTS. A sexta coluna mostra o desvio relativo percentual $dev(t) = (t^{HVNTS} - t^{VNS})/t^{HVNTS}$ entre a média de tempo de execução de VNS-VRPMTW (t^{VNS}) e a média de tempo de execução de HVNTS (t^{HVNTS}). Nessa métrica, valores positivos indicam que o tempo de execução de VNS-VRPMTW é menor que o tempo de execução de HVNTS.

Pode-se observar que VNS-VRPMTW consegue, em média, soluções $3,52\%$ melhores que o HVNTS. A heurística proposta é melhor em nove dos doze grupos de instâncias. Porém, a heurística proposta gasta, aproximadamente, $133,82\%$ mais tempo que HVNTS em média. Parte do bom desempenho de VNS-VRPMTW pode ser atribuido ao fato de que é capaz de eliminar mais rotas que HVNTS. Nos demais casos,

Tabela 4.6. Resultados médios de 10 execuções independentes das heurísticas VNS-VRPMTW e HVNTS com o uso da função objetivo (1.1)

Grupos	HVNTS		VNS-VRPMTW		%dev	
	f(S)	Tempo (s)	*f(S)*	Tempo (s)	dev(f) %	dev(t) %
rm1	2831,12	108,96	2789,13	389,26	1,48	-257,27
rm2	3525,06	107,95	2863,00	564,61	18,78	-423,02
cm1	3453,47	159,36	3568,88	175,10	-3,34	-9,88
cm2	5098,53	147,60	4331,63	240,71	15,04	-63,08
rcm1	3468,88	94,89	3340,75	222,04	3,69	-133,99
rcm2	3793,33	140,04	3603,63	386,90	5,00	-176,29
pr1	2695,59	111,46	2677,25	289,17	0,68	-159,43
pr2	2743,74	133,24	2704,25	455,93	1,44	-242,19
pc1	2867,45	143,49	2869,25	98,15	-0,06	31,60
pc2	2751,69	157,00	2733,50	366,27	0,66	-133,30
prc1	2810,87	101,85	2911,75	105,03	-3,59	-3,12
prc2	2772,08	181,55	2704,25	246,69	2,45	-35,89
				Média:	3,52	-133,82

a heurística proposta é tem desempenho semelhante, mas encontra melhores soluções. Isso provavelmente se dá pelo uso do VND proposto que também pode influenciar no maior gasto de tempo.

4.2 Resultados para a variante cuja função objetivo é (1.2)

O sétimo experimento realizado avalia o impacto das buscas locais baseadas nos operadores de vizinhança de múltiplas rotas no desempenho do procedimento VND executado na linha 5 do algoritmo 2. Neste experimento, foram utilizadas as sete variações de VNS-VRPMTW definidas na seção anterior: VNS-ALL, VNS-R, VNS-S, VNS-C, VNS-3N, VNS-3E e VNS-3NE. Neste experimento, essas heurísticas se diferem das utilizadas no primeiro experimento por usar a função objetivo (1.2) e não executar o procedimento MBTSA na linha 6 do algoritmo 2. As heurísticas foram executadas dez vezes para os grupos de instâncias pr1, pr2, pc1, pc2, prc1 e prc2.

A Tabela 4.7 apresenta os resultados desse experimento. Os grupos de instâncias são apresentados na primeira coluna. Na segunda e terceira colunas, são apresentadas as médias dos custos das soluções encontradas e dos tempos de execução de VNS-ALL respectivamente. A quarta coluna mostra o desvio relativo percentual $dev(f) = (f^{ALL} - f^{R})/f^{ALL}$ entre a média de custo das soluções obtidas por VNS-R (f^{R}) e a média de custo das soluções obtidas por VNS-ALL (f^{ALL}). Vale salientar que valores

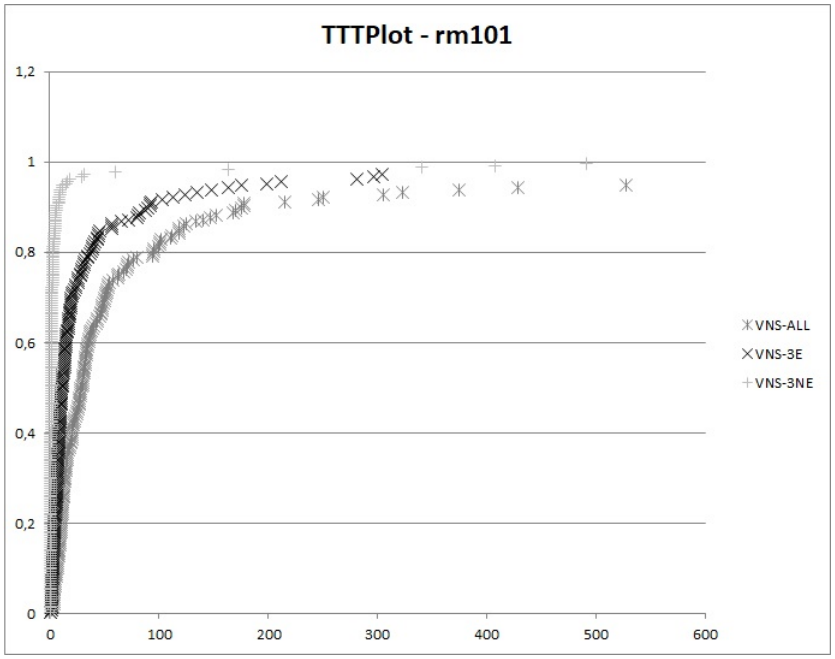

Figura 4.5. TTTPlot de 200 execuções das variantes VNS-ALL, VNS-3E e VNS-3NE para a instância rm101.

positivos para esta métrica indicam que as soluções de VNS-R são melhores que as soluções de VNS-ALL. A quinta coluna mostra o desvio relativo percentual $dev(t) = (t^{ALL} - t^R)/t^{ALL}$ entre a média de tempo de execução de VNS-R (t^R) e a média de tempo de execução de VNS-ALL (t^{ALL}). Vale salientar que valores positivos para esta métrica indicam que os tempos de VNS-R são menores que os tempos de VNS-ALL. Os mesmos valores são apresentados nas sexta e sétima colunas, respectivamente, para VNS-S, nas oitava e nona colunas, respectivamente, para VNS-C, nas décima e décima primeira colunas, respectivamente, para VNS-3N, nas décima segunda e décima terceira colunas, respectivamente, para VNS-3E e, finalmente, nas duas últimas colunas colunas para VNS-3NE.

Pode-se observar neste experimento que VNS-3N e VNS-3E encontraram soluções melhores que VNS-ALL. Por sua vez, VNS-3NE encontrou, em média, soluções $0,12\%$ piores que VNS-ALL, porém o tempo de execução de VNS-3NE é $63,65\%$ menor que o tempo de execução em VNS-ALL. Portanto, novamente, para verificar dentre as variantes VNS-ALL, VNS-3E e VNS-3NE qual é a de melhor desempenho, foi utilizada a técnica de verificação gráfica TTTPlot. O teste consistiu de duzentas execuções das variantes para um conjunto de instâncias e cada execução foi limitada a até seiscentos segundos de execução sem atingir o alvo.

Os resultados desse experimento para as instâncias rm101, cm101, pr101 e pc101

Tabela 4.7. Resultados médios de 10 execuções independentes das heurísticas VNS-ALL, VNS-R, VNS-S, VNS-C, VNS-3N, VNS-3E e VNS-3NE com o uso da função objetivo (1.2)

Grupos	VNS-ALL		VNS-R		VNS-S		VNS-C		VNS-3N		VNS-3E		VNS-3NE	
	f(S)	Tempo (s)	dev(f) %	dev(t) %	dev(f) %	dev(t) %	dev(f) %	dev(t) %	dev(f) %	dev(t) %	dev(f) %	dev(t) %	dev(f) %	dev(t) %
rm1	2865,75	241,46	-1,94	-53,36	0,23	-39,33	0,46	4,76	0,12	42,82	0,34	43,68	0,53	92,11
rm2	2856,25	49,76	-0,50	-0,95	-0,59	-9,42	-0,59	2,88	0,31	5,83	0,30	-76,06	0,27	-41,94
cm1	3340,75	341,47	-1,72	11,24	-0,20	-16,44	-0,71	-8,85	-0,89	56,16	-0,09	54,80	-0,76	95,92
cm2	4250,00	247,93	-1,49	-13,11	-0,10	-12,37	-0,12	-16,45	-0,07	38,90	-0,11	53,89	-0,14	90,44
rcm1	3360,88	213,91	-1,63	7,00	-0,03	-85,37	0,09	0,72	-0,08	45,28	-0,11	52,05	-0,41	93,00
rcm2	3684,25	124,47	-1,47	-13,68	-0,32	-57,46	-0,76	-0,63	-0,68	29,09	-0,07	38,80	-0,68	53,93
pr1	2671,50	577,71	-3,14	56,52	-0,60	37,05	-0,70	36,88	-0,27	44,08	-1,02	51,91	-1,13	93,50
pr2	2988,50	93,28	-0,25	37,55	1,53	2,66	-2,94	-29,80	2,16	-26,11	0,58	6,16	3,46	36,67
pc1	2825,50	238,39	-1,38	-128,54	-0,11	-25,13	-0,10	-14,03	-0,13	22,32	0,03	12,05	-0,04	96,43
pc2	2924,00	250,79	-4,87	-53,71	-1,86	-94,76	-1,53	-1,85	-0,68	-8,87	0,98	2,64	-5,03	49,23
prc1	2925,50	204,61	-3,56	-35,56	-1,62	-28,51	-0,07	1,21	-0,17	9,69	-0,85	28,05	-1,03	91,51
prc2	3404,75	205,41	-4,37	42,89	1,29	22,07	0,51	32,11	0,51	32,10	1,47	40,41	3,56	13,02
		Média	-2,19	-11,97	-0,20	-25,58	-0,54	0,58	0,01	24,28	0,12	25,70	-0,12	63,65

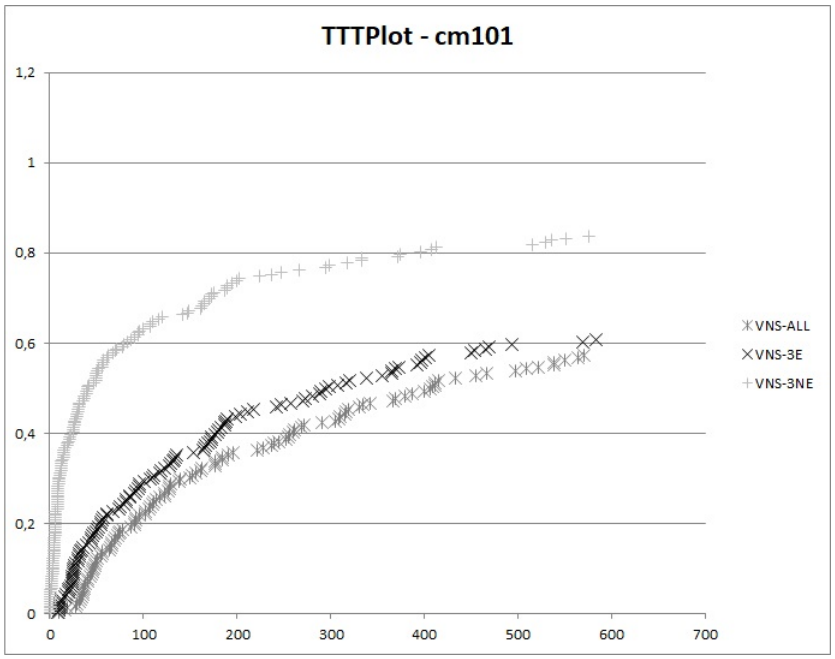

Figura 4.6. TTTPlot de 200 execuções das variantes VNS-ALL, VNS-3E e VNS-3NE para a instância cm101.

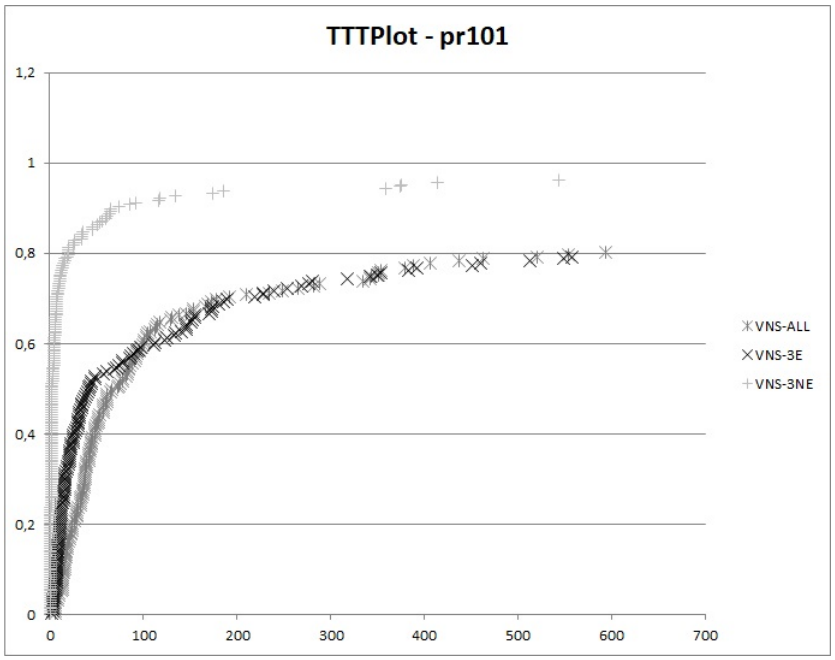

Figura 4.7. TTTPlot de 200 execuções das variantes VNS-ALL, VNS-3E e VNS-3NE para a instância pr101.

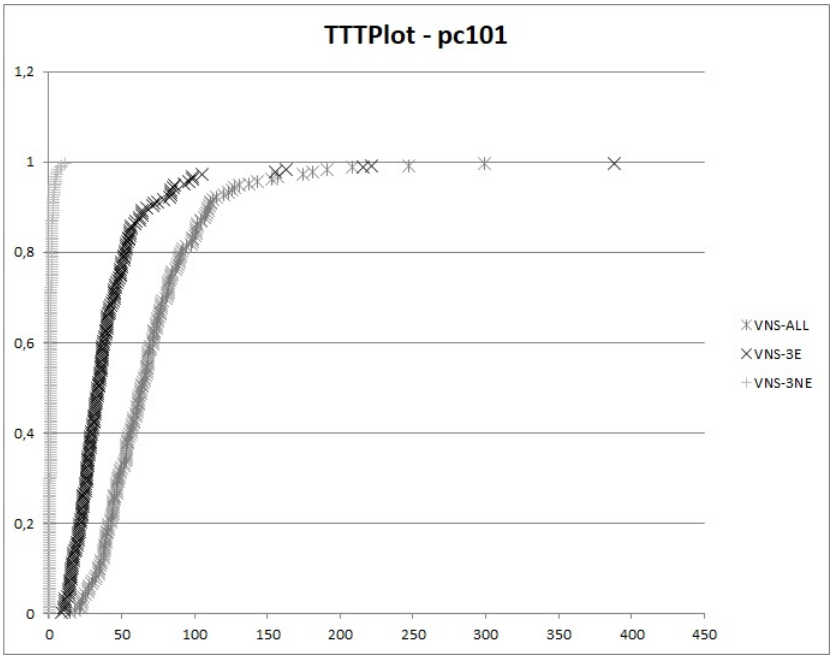

Figura 4.8. TTTPlot de 200 execuções das variantes VNS-ALL, VNS-3E e VNS-3NE para a instância pc101.

são apresentados nos gráficos nas Figuras 4.5, 4.6, 4.7 e 4.8. Pode ser observado que, nesse experimento, a variante VNS-3NE possui melhor desempenho do que as variantes VNS-3E e VNS-ALL em todos os casos. Devido a esse fato, a versão de VNS-VRPMTW utilizada no experimento a seguir é VNS-3NE, que não inclui as buscas locais que utilizam os operadores *multi-route 3-node swap* e *multi-route 3-exchange*.

O oitavo experimento realizado verifica o desempenho do procedimento VND executado na linha 5 do algoritmo 2 proposto nessa dissertação para a variante da heurística proposta que utiliza a função objetivo (1.2). Nesse experimento, foram utilizadas as três variantes de VNS-VRPMTW definidas na seção anterior: VNS-VNDP, VNS-VNDL e VNS-VNDR. As heurísticas foram executadas dez vezes para todos os grupos de instância.

Os resultados para esse experimento são apresentados na Tabela 4.8. Os grupos de instâncias são apresentadas na primeira coluna. Na segunda, terceira e quarta colunas, são apresentadas as médias dos custos das soluções encontradas, dos tempos de execução e do número total de iterações de VNS-VNDP respectivamente. A quinta coluna mostra o desvio relativo percentual $dev(f) = (f^{VNDP} - f^{VNDL})/f^{VNDP}$ entre a média de custo das soluções obtidas por VNS-VNDL (f^{VNDL}) e a média de custo das soluções obtidas por VNS-VNDP (f^{VNDP}). Vale salientar que valores positivos para esta métrica indicam que as soluções de VNS-VNDP são melhores

Tabela 4.8. Resultados médios de 10 execuções independentes das heurísticas VNS-VNDP, VNS-VNDL e VNS-VNDR com o uso da função objetivo (1.2)

Grupos	VNS - VNDP			VNS - VNDL			VNS - VNDR		
	f(S)	Tempo (s)	Tot. Iter	dev(f)%	dev(t)%	dev(i)%	dev(f)%	dev(t)%	dev(i)%
rm1	2850,88	18,15	1346,50	-8,59	14,86	6,07	-8,15	32,38	18,05
rm2	2847,25	50,78	1346,75	-10,97	-311,95	25,62	-8,90	-209,54	22,16
cm1	3366,88	13,07	1741,13	-21,25	71,46	7,24	-21,20	71,28	11,88
cm2	4255,00	22,57	1656,38	-20,33	71,21	20,30	-20,69	60,31	26,45
rcm1	3374,75	14,75	1308,38	-6,21	56,95	-8,33	-6,27	42,65	3,39
rcm2	3707,88	56,63	1327,25	-13,22	29,66	43,73	-13,37	18,16	45,53
pr1	2701,75	34,12	1279,25	-16,88	74,64	34,60	-15,90	67,94	29,34
pr2	2885,75	51,56	1576,25	-11,81	1,54	34,55	-15,06	-8,87	25,33
pc1	2826,50	8,37	1315,25	-18,73	-148,70	25,64	-18,33	-90,32	25,29
pc2	3069,50	124,24	1363,50	-24,39	18,68	50,23	-26,36	64,80	56,74
prc1	2955,00	17,31	1185,50	-15,10	23,45	20,80	-17,15	73,73	32,97
prc2	3282,50	181,17	1440,75	-18,54	63,24	32,50	-19,00	77,80	37,08
			Média	-15,50	-2,91	24,41	-15,86	16,69	27,85

que as soluções de VNS-VNDL. A sexta coluna mostra o desvio relativo percentual $dev(t) = (t^{VNDP} - t^{VNDL})/t^{VNDP}$ entre a média de tempo de execução de VNS-VNDL (t^{VNDL}) e de VNS-VNDP (t^{VNDP}). Vale salientar que valores positivos para esta métrica indicam que os tempos de execução de VNS-VNDP são menores que os tempos de execução de VNS-VNDL. A sétima coluna mostra o desvio relativo percentual $dev(i) = (i^{VNDP} - i^{VNDL})/i^{VNDP}$ entre a média do número de iterações de VNS-VNDL (i^{VNDL}) e o número de iterações de VNS-VNDP (i^{VNDP}). Vale salientar que valores positivos para esta métrica indicam que VNS-VNDP realiza menos iterações que VNS-VNDL. Os mesmos valores são apresentados na oitava, nona e décima colunas para VNS-VNDR.

Nesse caso, as soluções encontradas por VNS-VNDP são $15,50\%$ melhores que as soluções encontradas por VNS-VNDL e $15,86\%$ melhor que VNS-VNDR. Nessa variação, portanto, a heurística VNS-VNDP consegue escapar dos ótimos locais das outras buscas locais com mais facilidade que as outras variações. Essa variação na qualidade das soluções também pode ser explicada pela diferença no número de iterações de VNS-VNDP, sendo ele $24,41\%$ e $27,85\%$ maior que o número de iterações de VNS-VNDL e VNS-VNDR respectivamente. Esse maior número de iterações também implica na diferença de tempo de execução entre as heurísticas. Portanto, a versão escolhida para ser utilizada nos demais experimentos foi VNS-VNDP.

O nono experimento realizado procura definir o melhor critério de parada para VNS-VRPMTW quando utiliza-se a função objetivo (1.2). Para tanto, foram propostas três variantes de VNS-VRPMTW. As variantes, VNS-3000, VNS-4000 e VNS-5000, têm como critério de parada do *loop* na linha 3 do algoritmo 2 3000, 4000 e 5000

Tabela 4.9. Resultados médios de 10 execuções independentes das heurísticas VNS-3000, VNS-4000 e VNS-5000 com o uso da função objetivo (1.2) em comparação com os resultados de HVNTS

Grupos	HVNTS		VNS-3000		VNS-4000		VNS-5000	
	f(S)	Tempo (s)	dev(f) %	dev(t) %	dev(f) %	dev(t) %	dev(f) %	dev(t) %
rm1	2805,88	91,11	-1,30	34,31	-0,26	37,49	0,03	27,39
rm2	3473,38	95,60	15,82	-19,86	15,98	-25,13	15,89	-53,71
cm1	3345,00	101,50	0,77	53,99	1,29	63,28	1,23	42,39
cm2	5045,38	199,22	15,98	37,47	15,98	50,53	16,27	38,74
rcm1	3411,63	248,14	2,17	55,04	2,76	63,12	3,10	47,48
rcm2	3710,50	91,76	1,66	9,80	1,03	22,34	0,97	11,01
prm1	2814,50	92,56	5,09	59,93	5,34	56,14	5,28	34,50
prm2	2756,25	108,53	-7,00	14,70	-6,82	2,38	-6,02	-17,31
pcm1	2834,50	87,57	0,37	80,19	0,37	78,97	0,37	73,39
pcm2	2742,00	96,59	-2,24	-98,20	-0,27	-149,10	-4,35	-229,45
prcm1	2680,00	80,97	-7,86	73,71	-6,71	67,56	-6,78	52,47
prcm2	2728,50	110,03	-17,13	-3,56	-18,10	-13,47	-12,82	-63,14
		Média:	0,53	24,79	0,88	21,18	1,10	-3,02

iterações sem melhora da melhor solução conhecida respectivamente. Diferentemente das variantes que utilizam a função objetivo (1.2), os valores utilizados no critério de parada de VNS-3000, VNS-4000 e VNS-5000 são maiores por não haver o uso das buscas locais que utilizam as vizinhanças *multi-route 3 node swap* e *multi-route 3 exchange*. As heurísticas foram executadas dez vezes para todos os grupos de instâncias e comparadas com os resultados de 10 execuções de HVNTS conforme proposto em [Belhaiza et al., 2014].

Os resultados desse experimento são apresentados na Tabela 4.9. Na primeira coluna, são apresentados os grupos de instâncias usados. Na segunda e terceira, são apresentadas as médias dos custos das melhores soluções encontradas e dos tempos de execução para dez execuções independentes de HVNTS. Na quarta coluna é apresentado o desvio relativo percentual $dev(f) = (f^{HVNTS} - f^{3000})/f^{HVNTS}$ entre a média de custo das soluções obtidas por VNS-100 (f^{3000}) e a média de custo das soluções obtidas por HVNTS (f^{HVNTS}). Vale salientar que valores positivos para esta métrica indicam que as soluções de VNS-100 são melhores que as soluções de HVNTS. A quinta coluna mostra o desvio relativo percentual $dev(t) = (t^{HVNTS} - t^{3000})/t^{HVNTS}$ entre a média de tempo de execução de VNS-100 (t^{3000}) e a média de tempo de execução de HVNTS (t^{HVNTS}). Nessa métrica, valores positivos indicam que o tempo de execução de VNS-VRPMTW é menor que o tempo de execução de HVNTS. Os mesmos valores são apresentados nas sexta e sétima colunas para VNS-4000 e nas oitava e nona colunas para VNS-5000.

Pode ser observado que, também nesse experimento, as três variantes encontram

em média soluções de custo melhor que as encontradas pelo HVNTS. Mais uma vez, também, esses valores são influenciados pelos grupos rm2 e cm2, pois as heurísticas encontram soluções com um veículo a menos do que as soluções encontradas por HVNTS. Das três variantes, VNS-3000 e VNS-4000 encontraram melhores soluções em sete dos doze grupos de instâncias, sendo que o desempenho de VNS-4000 é semelhante ao desempenho de VNS-5000. Portanto, a variante escolhida para ser usada no restante dos testes foi VNS-5000, pois ela encontra melhores soluções em oito dos doze grupos e porque o tempo de execução, em média, é próximo ao tempo gasto por HVNTS.

O décimo experimento compara os resultados de VNS-VRPMTW com aqueles de HVNTS para versão de VRPMTW cuja função objetivo é a (1.2). As duas heurísticas foram executadas 10 vezes sobre todos os grupos de instâncias. Os resultados desse experimento são apresentados na Tabela 4.10. Na primeira coluna, são apresentados os grupos de instâncias usados. Na segunda e terceira, são apresentadas as médias dos custos das melhores soluções encontradas e dos tempos de execução para dez execuções independentes de HVNTS. Na quarta e quinta, são apresentadas as médias dos custos das melhores soluções encontradas e dos tempos de execução para dez execuções independentes de VNS-VRPMTW. Na sexta coluna é apresentado o desvio relativo percentual $dev(f) = (f^{HVNTS} - f^{VNS})/f^{HVNTS}$ entre a média de custo das soluções obtidas por VNS-VRPMTW (f^{VNS}) e a média de custo das soluções obtidas por HVNTS (f^{HVNTS}). Vale salientar que valores positivos para esta métrica indicam que as soluções de VNS-VRPMTW são melhores que as soluções de HVNTS. A sexta coluna mostra o desvio relativo percentual $dev(t) = (t^{HVNTS} - t^{VNS})/t^{HVNTS}$ entre a média de tempo de execução de VNS-VRPMTW (t^{VNS}) e a média de tempo de execução de HVNTS (t^{HVNTS}). Nessa métrica, valores positivos indicam que o tempo de execução de VNS-VRPMTW é menor que o tempo de execução de HVNTS.

Observa-se que a heurística proposta consegue, em média, soluções $1,1\%$ melhores que o HVNTS. Ela é melhor em oito dos doze grupos de instâncias. O tempo de execução é, em média, $3,02\%$ maior que o tempo de execução de HVNTS. Esse valor foi mais influenciado por pelo grupo de instâncias pc2, onde o tempo gasto foi $229,45\%$ maior que o gasto por HVNTS. Em sete dos oito casos em que a heurística proposta encontra melhores soluções que o HVNTS, o tempo de execução é, pelo menos 11% menor que o tempo gasto pelo HVNTS. Isso demonstra que as vizinhanças *multi-route 3 node swap* e *multi-route 3 exchange* não tem muito impacto na otimização utilzando desses grupos de instâncias.

Tabela 4.10. Resultados médios de 10 execuções independentes das heurísticas VNS-VRPMTW e HVNTS com o uso da função objetivo (1.2)

Grupos	HVNTS		VNS-VRPMTW		%dev	
	f(S)	Tempo (s)	*f(S)*	Tempo (s)	dev(f) %	dev(t) %
rm1	2805,88	91,11	2805,00	53,73	0,03	27,39
rm2	3473,38	95,60	2864,50	128,96	15,89	-53,71
cm1	3345,00	101,50	3327,38	62,40	1,23	42,39
cm2	5045,38	199,22	4223,63	64,39	16,27	38,74
rcm1	3411,63	248,14	3305,88	49,11	3,10	47,48
rcm2	3710,50	91,76	3674,38	82,01	0,97	11,01
pr1	2814,50	92,56	2666,00	59,92	5,28	34,50
pr2	2756,25	108,53	2924,00	127,30	-6,02	-17,31
pc1	2834,50	87,57	2824,00	22,60	0,37	73,39
pc2	2742,00	96,59	2861,00	317,23	-4,35	-229,45
prc1	2680,20	80,97	2861,75	38,37	-6,78	52,47
prc2	2728,50	110,03	3078,00	175,24	-12,82	-63,14
				Média	1,10	-3,02

Capítulo 5

Conclusão

Neste trabalho, foi proposta uma heurística para resolução de VRPMTW. A heurística, chamada de VNS-VRPMTW, foi desenvolvida de forma a considerar apenas direções viáveis, isso é, movimentos que respeitem a todas as restrições do problema. Ela se divide em três fases. Na primeira, a solução inicial é gerada por uma heurística construtiva baseada na heurística PFIH [Solomon, 1987]. Na segunda, a heurística *Route-Elimination* é usada para a minimização do número de rotas da solução inicial. Na fase subsequente, é feita a minimização do tempo total da solução através de uma herística baseada em VNS chamada de *Route-Optimization*. Nessa heurística, também foi utilizado o algoritmo MBTSA [Belhaiza et al., 2014] para a minimização do tempo de espera.

A heurística foi então comparada com a melhor heurística na literatura de VRPMTW. Essa heurística, chamada de *Hybrid Variable Neighborhood Tabu Search* (HVNTS), permite movimentos que gerem soluções inviáveis desde que uma penalidade seja adicionada ao custo da solução. Experimentos computacionais foram executados para duas variantes de VRPMTW. Na primeira variante, o tempo de espera em cada cliente é considerado para o cálculo do custo de uma solução, enquanto na segunda variante, o tempo de espera não é considerado na função objetivo.

Experimentos computacionais mostraram que nem todos os operadores de vizinhança propostos por Belhaiza et al. [2014] são computacionalmente eficientes em VNS-VRPMTW. Além disso, eles mostram que avaliar somente os custos de vizinhos cujo tempo de viagem é menor ou igual àquele da solução corrente é mais vantajoso que avaliar o custo de todo vizinho possível. Também verificou-se que o uso de um algoritmo para redução do tempo de espera em uma rota é necessário para que se encontre melhores soluções.

Por fim, foi feito um experimento para determinar o desempenho da heurística em

relação a HVNTS. A heurística proposta encontra, em média, soluções $3,52\%$ melhores que as encontradas por HVNTS para quando o tempo de espera é considerado na otimização do problema. O tempo gasto pela heurística, nesse caso, é maior que o de HVNTS, porém isso pode ser corrigido com uso de uma máquina melhor ou melhores técnicas de implementação. Quando o tempo de espera não é considerado na função objetivo, a heurística proposta encontra, em média, soluções $1,1\%$ melhores e tem tempo de execução $3,02\%$ menor que o de HVNTS.

Dessa forma, demonstra-se que, com apenas movimentos viáveis, é possível obter soluções tão boas quanto aquelas encontradas por HVNTS. Este resultado indica que diferentes regiões do espaço de busca, onde são encontradas boas soluções, podem ser alcançadas sem passar por soluções inviáveis. Isso é um indício de que o mesmo cenário pode ser encontrado em outras situações, o que podem orientar o projeto de pesquisas futuras sobre VRPMTW.

Uma vez que VNS-VRPMTW sempre gera soluções viáveis e o custo das soluções de VRPMTW consiste na soma do custo individual de cada rota, VNS-VRPMTW pode ser empregada na abordagem utilizada por Monaci & Toth [2006] para Bin packing, Kelly & Xu [1999] para VRP, Morais et al. [2014] para VRP com Cross-docking e Alvarenga et al. [2007] para VRPTW. No caso de VRPMTW, as rotas de todas as soluções viáveis encontradas por VNS-VRPMTW podem ser identificadas e utilizadas em uma formulação de programação linear inteira para o *Weighted Set Covering Problem* [Edmonds, 1962], onde o conjunto universo é igual ao conjunto de clientes, e cada rota cobre um subconjunto de clientes. Desta forma, um algoritmo que resolve esta formulação pode encontrar soluções ainda melhores que VNS-VRPMTW ao combinar de forma ótima as rotas presentes em qualquer das soluções encontradas por VNS-VRPMTW.

Referências Bibliográficas

Ahuja, R. K.; Orlin, J. B. & Sharma, D. (2000). Very large-scale neighborhood search. *International Transactions in Operational Research*, 7(4-5):301--317.

Alvarenga, G. B. (2005). *Um algoritmo híbrido para o problema de roteamento de veículos com janelas de tempo*. Tese de doutorado, Universidade Federal de Minas Gerais, MG.

Alvarenga, G. B.; Mateus, G. R. & De Tomi, G. (2007). A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows. *Computers & Operations Research*, 34(6):1561--1584.

Baldacci, R.; Hadjiconstantinou, E. & Mingozzi, A. (2004). An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. *Operations research*, 52(5):723--738.

Balinski, M. L. & Quandt, R. E. (1964). On an integer program for a delivery problem. *Operations Research*, 12(2):300--304.

Belhaiza, S.; Hansen, P. & Laporte, G. (2014). A hybrid variable neighborhood tabu search heuristic for the vehicle routing problem with multiple time windows. *Computers & Operations Research*, 52:269--281.

Bent, R. & Van Hentenryck, P. (2004). A two-stage hybrid local search for the vehicle routing problem with time windows. *Transportation Science*, 38(4):515--530.

Berbeglia, G.; Cordeau, J.-F.; Gribkovskaia, I. & Laporte, G. (2007). Static pickup and delivery problems: a classification scheme and survey. *Top*, 15(1):1--31.

Berger, J.; Barkaoui, M. & Bräysy, O. (2003). A route-directed hybrid genetic approach for the vehicle routing problem with time windows. *INFOR: Information Systems and Operational Research*, 41(2):179--194.

Bräysy, O. (2003). A reactive variable neighborhood search for the vehicle-routing problem with time windows. *INFORMS Journal on Computing*, 15(4):347--368.

Bräysy, O. & Gendreau, M. (2005a). Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation science*, 39(1):104--118.

Bräysy, O. & Gendreau, M. (2005b). Vehicle routing problem with time windows, part ii: Metaheuristics. *Transportation science*, 39(1):119--139.

Christiansen, M. & Fagerholt, K. (2002). Robust ship scheduling with multiple time windows. *Naval Research Logistics (NRL)*, 49(6):611--625.

CNT (2016). Custo logístico consome 12,7% do pib do brasil. *Disponível em* `http://www.cnt.org.br/Imprensa/noticia/custo-logistico-consome-12-do-pib-do-brasil`. *Acessado em: 01/Nov/2017.*

Dantzig, G.; Fulkerson, R. & Johnson, S. (1954). Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America*, 2(4):393--410.

Dantzig, G. B. & Ramser, J. H. (1959). The truck dispatching problem. *Management science*, 6(1):80--91.

David Houck, J. (1978). *The traveling salesman problem as a constrained shortest path problem: Theory and computational experience*. Ecole polytechnique de Montréal.

de Jong, C.; Kant, G. & Van Vlient, A. (1996). On finding minimal route duration in the vehicle routing problem with multiple time windows. *Manuscript, Department of Computer Science, Utrecht University, Holland.*

Edmonds, J. (1962). Covers and packings in a family of sets. *Bulletin of the American Mathematical Society*, 68:494–499.

Favaretto, D.; Moretti, E. & Pellegrini, P. (2007). Ant colony system for a vrp with multiple time windows and multiple visits. *Journal of Interdisciplinary Mathematics*, 10(2):263--284.

Feo, T. A. & Resende, M. G. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations research letters*, 8(2):67--71.

Gehring, H. & Homberger, J. (1999). A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. Em *Proceedings of EUROGEN99*, volume 2, pp. 57--64. Citeseer.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5):533--549.

Golden, B. L.; Assad, A. A. & Wasil, E. A. (2001). Routing vehicles in the real world: applications in the solid waste, beverage, food, dairy, and newspaper industries. Em *The vehicle routing problem*, pp. 245--286. Society for Industrial and Applied Mathematics.

Held, M. & Karp, R. M. (1970). The traveling-salesman problem and minimum spanning trees. *Operations Research*, 18(6):1138--1162.

Held, M. & Karp, R. M. (1971). The traveling-salesman problem and minimum spanning trees: Part ii. *Mathematical programming*, 1(1):6--25.

Holland, J. H. (1992). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press.

Homberger, J. & Gehring, H. (2005). A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. *European Journal of Operational Research*, 162(1):220--238.

Ibaraki, T.; Imahori, S.; Kubo, M.; Masuda, T.; Uno, T. & Yagiura, M. (2005). Effective local search algorithms for routing and scheduling problems with general time-window constraints. *Transportation science*, 39(2):206--232.

Ioannou, G.; Kritikos, M. & Prastacos, G. (2001). A greedy look-ahead heuristic for the vehicle routing problem with time windows. *Journal of the Operational Research Society*, 52(5):523--537.

Kallehauge, B. (2008). Formulations and exact algorithms for the vehicle routing problem with time windows. *Computers & Operations Research*, 35(7):2307--2330.

Kelly, J. P. & Xu, J. (1999). A set-partitioning-based heuristic for the vehicle routing problem. *INFORMS Journal on Computing*, 11(2):161--172.

Kirkpatrick, S.; Gelatt, C. D.; Vecchi, M. P. et al. (1983). Optimization by simulated annealing. *science*, 220(4598):671--680.

Laporte, G. (2007). What you should know about the vehicle routing problem. *Naval Research Logistics (NRL)*, 54(8):811--819.

Laporte, G.; Nobert, Y. & Desrochers, M. (1985). Optimal routing under capacity and distance restrictions. *Operations research*, 33(5):1050--1073.

Lin, S. & Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, 21(2):498--516.

Lourenço, H. R.; Martin, O. C. & Stutzle, T. (2003). Iterated local search. *International series in operations research and management science*, pp. 321--354.

Miller, C. E.; Tucker, A. W. & Zemlin, R. A. (1960). Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 7(4):326--329.

MINC (2013). I inventário nacional de emissões atmosféricas por veículos automotores rodoviários. *Disponível em: http://www.mma.gov.br/estruturas/182/_arquivos/inventorio_de_emisses_veiculares_182.pdf. Acessado em: 11/Nov/2017.*

Mladenović, N. & Hansen, P. (1997). Variable neighborhood search. *Computers & operations research*, 24(11):1097--1100.

Monaci, M. & Toth, P. (2006). A set-covering-based heuristic approach for bin-packing problems. *INFORMS Journal on Computing*, 18(1):71--85.

Morais, V. W.; Mateus, G. R. & Noronha, T. F. (2014). Iterated local search heuristics for the vehicle routing problem with cross-docking. *Expert Systems with Applications*, 41(16):7495--7506.

Nagata, Y. & Bräysy, O. (2009). A powerful route minimization heuristic for the vehicle routing problem with time windows. *Operations Research Letters*, 37(5):333--338.

Nagata, Y.; Bräysy, O. & Dullaert, W. (2010). A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research*, 37(4):724--737.

Pesant, G.; Gendreau, M.; Potvin, J.-Y. & Rousseau, J.-M. (1998). An exact constraint logic programming algorithm for the traveling salesman problem with time windows. *Transportation Science*, 32(1):12--29.

Pesant, G.; Gendreau, M.; Potvin, J.-Y. & Rousseau, J.-M. (1999). On the flexibility of constraint programming models: From single to multiple time windows for the

traveling salesman problem. *European Journal of Operational Research*, 117(2):253--263.

Raff, S. (1983). Routing and scheduling of vehicles and crews: The state of the art. *Computers & Operations Research*, 10(2):63--211.

Renaud, J.; Laporte, G. & Boctor, F. F. (1996). A tabu search heuristic for the multi-depot vehicle routing problem. *Computers & Operations Research*, 23(3):229--235.

Savelsbergh, M. W. (1985). Local search in routing problems with time windows. *Annals of Operations research*, 4(1):285--305.

Savelsbergh, M. W. & Sol, M. (1995). The general pickup and delivery problem. *Transportation science*, 29(1):17--29.

Schrimpf, G.; Schneider, J.; Stamm-Wilbrandt, H. & Dueck, G. (2000). Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, 159(2):139--171.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35(2):254--265.

Solomon, M. M. & Desrosiers, J. (1988). Survey paper-time window constrained routing and scheduling problems. *Transportation science*, 22(1):1--13.

Toth, P. & Vigo, D. (2014). *Vehicle routing: problems, methods, and applications*, volume 18. Siam.

Tricoire, F.; Romauch, M.; Doerner, K. F. & Hartl, R. F. (2010). Heuristics for the multi-period orienteering problem with multiple time windows. *Computers & Operations Research*, 37(2):351--367.