ALGORITMOS PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM CROSS-DOCKING

VINÍCIUS WELLINGTON COELHO DE MORAIS

ALGORITMOS PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM CROSS-DOCKING

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

Orientador: Geraldo Robson Mateus

Coorientador: Thiago Ferreira de Noronha

Belo Horizonte

Dezembro de 2012

Morais, Vinícius Wellington Coelho de
M827a Algoritmos para o Problema de Roteamento de Veículos com Cross-Docking / Vinícius Wellington Coelho de Morais. — Belo Horizonte, 2012
xxvi, 53 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais - Departamento de Ciência da Computação
Orientador: Geraldo Robson Mateus
Coorientador: Thiago Ferreira de Noronha

1. Computação - Teses. 2. Otimização Combinatória – Teses. 3. I. Orientador. 4. II.Coorientador. I. Título.

CDU 519.6*61 (043)

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

FOLHA DE APROVAÇÃO

Algoritmos para o problema de roteamento de veículos com cross-docking

VINÍCIUS WELLINGTON COELHO DE MORAIS

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. GERALDO ROBSON MATEUS - Orientador
Departamento de Ciência da Computação - UFMG

PROF. THIAGO FERREIRA DE NORONHA - Coorientador
Departamento de Ciência da Computação - UFMG

PROF. GILBERTO DE MIRANDA JÚNIOR
Departamento de Engenharia de Produção - UFMG

PROF. HAROLDO GAMBINI SANTOS
Departamento de Computação - UFOP

Belo Horizonte, 18 de dezembro de 2012.

Dedico este trabalho à minha mãe Cida, à minha Vó Ana, aos meus irmãos Vanessa, Vivian e Álvaro e ao meu padastro Alair.

Agradecimentos

Esta dissertação é fruto de vinte meses de trabalho. Neste período várias pessoas contribuíram para o seu desenvolvimento. Gostaria de agradecer a todos pelo apoio.

Agradeço à minha mãe, minha principal incentivadora. Aos meus irmãos, Vanessa, Vivian e Álvaro, que sempre foram a inspiração para meus esforços. A todos os meus familiares tio(as), primos(as), meu padastro e meus avós. Estes contribuíram diretamente para minha formação, caráter e personalidade.

Aos meus amigos e colegas das repúblicas de Diamantina e Belo Horizonte: Drika, Danilo, Lucas Costa, Lucas Meirelles, Jucimara, Henrique, Tiago e Michelle. Sou grato por me receberem como família. Agradeço à Gigi e todos aqueles que estiveram comigo nos momentos de felicidade, tristeza e descontração. Obrigado pelo carinho!

Aos amigos dos departamentos DECOM/UFVJM e DCC/UFMG. Aos amigos do LaPO/UFMG: Vitor Shiryu, Ramon, Rangel, Fernanda, Fernando, Gustavo, Januário, Afonso, Amadeu, Franklin, Phillippe Samer e Luis Henrique. Estes dividiram comigo o ambiente de trabalho e contribuíram diretamente com meu crescimento acadêmico e profissional.

Aos meus orientadores. Professores Alessandro Vivas e Luciana Assis, na graduação, e Geraldo Robson e Thiago Noronha, na pós-graduação. Agradeço pela disposição, acompanhamento e orientação. Sou grato pela paciência, dedicação, ensinamentos e pelas respostas positivas aos meus trabalhos.

Aos funcionários do DCC/UFMG, em especial as meninas da secretaria. Estes possibilitam o desenvolvimento deste trabalho.

Finalmente, agradeço à CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), pela concessão da bolsa durante todo o período de realização deste mestrado.

Resumo

Esta dissertação aborda o Problema de Roteamento de Veículos com *Cross-Docking*. Dados um conjunto de requisições por produtos, um único centro de consolidação (CD), e múltiplos fornecedores e consumidores, o problema consiste em definir rotas de custo mínimo para uma frota homogênea de veículos que realiza o transporte de produtos dos fornecedores aos consumidores via CD. Inicialmente, os veículos partem do CD em direção aos fornecedores, coletam os produtos e retornam ao ponto inicial para realizar o processo de consolidação. Em seguida, a mesma frota de veículos transporta os produtos aos consumidores. As rotas definidas devem respeitar as restrições de capacidade dos veículos e janela de tempo de cada fornecedor e consumidor. São propostas três heurísticas construtivas, uma heurística baseada na metaheurística Iterated Local Search e uma matheurística que integra esta última e um modelo de Programação Linear Inteira baseado em Particionamento de Conjuntos. Os algoritmos propostos foram avaliados em instâncias da literatura e comparados com as melhores heurísticas existentes para este problema. Os resultados obtidos mostram a eficiência das soluções implementadas.

Palavras-chave: Roteamento de Veículos, *Cross-Docking*, Heurística, Matheurística, Otimização Combinatória.

Abstract

This work addresses the Vehicle Routing Problem with Cross-Docking. Given a set of goods requests, a single Cross-Dock (CD), and several suppliers and customers, the problem consists in defining minimum cost set of routes for a fleet of identical vehicles that leaves the CD towards the suppliers, picks up the goods and returns to the initial point to perform the consolidation process. Thus, the same fleet of vehicles delivers the goods to the consumers. The routes designed must respect the capacity constraints of the vehicles as well as the time windows constraints. We proposed three constructive heuristics, a heuristic based on the Iterated Local Search metaheuristic, and a matheuristic that integrates the latter and a Integer Linear Programming model based on Set Partitioning. The algorithms were evaluated on instances from the literature and compared to the best existing heuristics for this problem. The results showed the efficiency of the implemented solutions.

Keywords: Vehicle Routing, Cross-Docking, Heuristic, Matheuristic, Combinatorial Optimization

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Siglas

2S-NI	*Two-Side Nearest Insertion*
2S-NN	*Two-Side Nearest Neighbor*
2S-S	*Two-Side Savings*
ALNS	*Adaptive Large Neighbourhood Search*
BCP	*Branch-and-Cut-and-Price*
CD	*Cross-Dock*
CRR	Conjunto de Requisições Candidatas
CVRP	*Capacitaded Vehicle Routing Problem*
DVRP	*Distance Constrained Vehicle Routing Problem*
EDP	*Express Delivery Problem*
HFVRP	*Heretogeneous Fleet Vehicle Routing Problem*
ILS	*Iterated Local Search*
MIP	*Mixed Integer Programming*
NI	*Nearest Insertion*
PDP	*Pickup and Delivery Problem*
SA	*Simulated Annealing*
SCM	*Supply Chain Management*
SP	*Set Partitioning*
TS	*Tabu Search*

TSP *Traveling Salesman Problem*

TTT plot *Time-To-Target value plots*

UTS *Unified Tabu Search*

VND *Variable Neighborhood Descent*

VRP *Vehicle Routing Problem*

VRPB *Vehicle Routing Problem with Backhaul*

VRPCD *Vehicle Routing Problem with Cross-Docking*

VRPPD *Vehicle Routing Problem with Pickup and Delivery*

VRPSD *Vehicle Routing with Split Delivery*

VRPSPD *Vehicle Routing Problem with Simultaneous Pickups and Deliveries*

VRPTW *Vehicle Routing Problem with Time Windows*

Sumário

Capítulo 1

Introdução

Devido à globalização e ao aumento da produtividade industrial e agrícola, o sistema de distribuição de produtos está cada vez mais em evidência. Com o objetivo de (i) minimizar o tempo de espera por produtos e serviços, (ii) elevar os índices de satisfação dos consumidores, (iii) reduzir custos e consequentemente (iv) maximizar os ganhos, soluções eficientes são aplicadas em atividades logísticas. A partir destas necessidades surge o conceito de Gestão da Cadeia de Suprimentos (SCM, do inglês *Supply Chain Management*).

A Cadeia de Suprimentos é o processo no qual diversas entidades de negócios (por exemplo, fornecedores, fabricantes, distribuidores e varejistas) integram suas ações para proporcionar uma melhor gestão do fluxo físico de materiais em uma rede logística. Esta também pode ser considerada sob uma rede de atividades responsáveis pela coleta, produção, armazenamento e transporte de mercadorias. Estas atividades possuem um certo grau de independência, porém a otimização de suas funções está diretamente vinculada ao desempenho da cadeia logística como um todo. Assim, o processo de otimização não deve ser visto apenas em função de uma única atividade [Jayaraman & Ross, 2003].

Segundo Chen et al. [2006],*Cross-docking* é um importante conceito vinculado ao armazenamento e gerenciamento do fluxo de produtos em um sistema de transporte. Nesta abordagem um conjunto de produtos vindos de diferentes fornecedores chega a um centro de consolidação (CD, do inglês *Cross-Dock*), onde as cargas são realocadas a uma frota de veículos e enviadas aos consumidores finais de forma a minimizar os custos de transporte. O termo *consolidação* é usado para se referir ao processo de agrupamento, preparo, carga e descarga de produtos em um CD [Napolitano, 2002]. Este processo é considerado de curto prazo, uma vez que os produtos não permanecem estocados por um período superior a um dia de trabalho [Yu & Egbelu, 2008].

Existe um grande número de trabalhos na literatura relacionados a *Cross-Docking* [Ratliff et al., 1999; Apte & Viswanathan, 2003; Bookbinder, 2004; Bartholdi & Gue, 2004]. Estes se restringem à definir a localização dos centros de consolidação ou planejar o sequenciamento dos veículos no CD. Entretanto, poucos são os trabalhos que tratam do processo de consolidação integrado com a tarefa de roteamento dos veículos entre os fornecedores/consumidores e o CD [Liao et al., 2010]. A partir desta observação Lee et al. [2006] definiram o Problema de Roteamento de Veículos com *Cross-Docking* (VRPCD, do inglês *Vehicle Routing Problem with Cross-Docking*).

No VRPCD são dados um único centro de consolidação, um conjunto de requisições, múltiplos fornecedores e consumidores, em que cada requisição está associada a um par fornecedor/consumidor. O problema consiste em minimizar os custos de transporte ao se atribuir rotas de coleta e entrega para uma frota de veículos que comecem e terminem no CD e que respeitem as restrições de capacidade dos veículos e janela de tempo em cada ponto da rede. Inicialmente, os veículos partem de sua origem em direção aos fornecedores, coletam suas requisições e retornam ao ponto inicial para realizar o processo de consolidação. Em seguida, a mesma frota de veículos entrega as requisições aos consumidores. Este problema é descrito formalmente a seguir.

1.1 Definição do Problema

Nesta dissertação centra-se no VRPCD, assim como descrito em Wen et al. [2009]. Este pode ser definido em um grafo $G = (N,E)$, em que o conjunto de nós é dado por $N = P \cup D \cup \{0\}$, onde $P = \{p_1, p_2,..., p_n\}$ é um conjunto de fornecedores, $D = \{d_1, d_2,..., d_n\}$ é um conjunto de consumidores e 0 um CD. Seja $E = \{(i,j) \mid i,j \in P\} \cup \{(i,j) \mid i,j \in D\} \cup \{(0,j) \mid j \in \{P \cup D\}\}$ um conjunto de arestas e $R = \{r_1, r_2,..., r_n\}$ um conjunto com n requisições, em que cada requisição $r_i \in R$ de um produto q_i é associada a um par $\{p_i, d_i\}$, tal que, $p_i \in P$ e $d_i \in D$. Seja ainda, uma frota V de veículos com capacidade limitada Q. A cada requisição $r_i \in R$ é associada uma demanda positiva de produtos σ_i, a cada aresta $(i,j) \in E$ é associado um custo c_{ij} (em tempo) e, para cada fornecedor $p_i \in P$ (ou consumidor $d_i \in D$) é associado uma janela de tempo $[a_i, b_i]$, onde a_i e b_i representam o tempo mínimo e máximo de chegada do veículo ao nó $i \in N$. O VRPCD consiste em definir rotas de coleta e entrega que minimizem o tempo total trafegado pela frota de veículos, em que os produtos de cada requisição $r_i \in R$ são coletados em um fornecedor $p_i \in P$ e entregues a um consumidor $d_i \in D$ passando pelo CD, onde o processo de consolidação é realizado.

As restrições do problema são as seguintes. (i) Tanto as rotas de coleta quanto

as rotas de entrega devem iniciar e terminar no CD. (ii) Cada fornecedor $p_i \in P$ (ou consumidor $d_i \in D$) deve ser visitado uma única vez. (iii) A mesma frota de veículos deve ser usada no processo de coleta e entrega. (iv) O somatório das demandas transportadas, tanto em uma rota de coleta quanto em uma rota de entrega, não deve extrapolar a capacidade do veículo. (v) Cada fornecedor (ou consumidor) deve ser visitado dentro de sua janela de tempo. (vi) As etapas de roteamento e consolidação devem ocorrer dentro do horizonte de planejamento T.

O tempo gasto pelo processo de consolidação é variável e é dado em função da quantidade de demanda por uma requisição, da penalidade imposta para carregar ou descarregar cada uma unidade de produto no CD mais um tempo fixo de preparo do veículo. Por exemplo, seja B o tempo para carregar (ou descarregar) cada unidade de produto no CD e A o tempo fixo de preparo do veículo, se A e B forem iguais a 10 e 2 minutos, respectivamente, e um veículo precisar consolidar os produtos de uma requisição com demanda de 22 unidades, então o tempo gasto para consolidar a requisição será de $22 \times 2 + 10 = 54$ minutos.

A Figura 1.1 ilustra o processo de roteamento do VRPCD. São mostrados dois pares de rotas para atender as demandas de cinco requisições, atribuídas aos veículos v_1 e v_2. As linhas tracejadas representam as rotas do veículo v_1, enquanto as linhas continuas retratam as rotas do veículo v_2. O veículo v_1 coleta os produtos das requisições r_4, r_1 e r_5 nos fornecedores e entrega os produtos das requisições r_2 e r_4 aos seus consumidores, ao mesmo tempo em que o veículo v_2 coleta os produtos das requisições r_3 e r_2 e entrega os produtos das requisições r_1, r_3 e r_5.

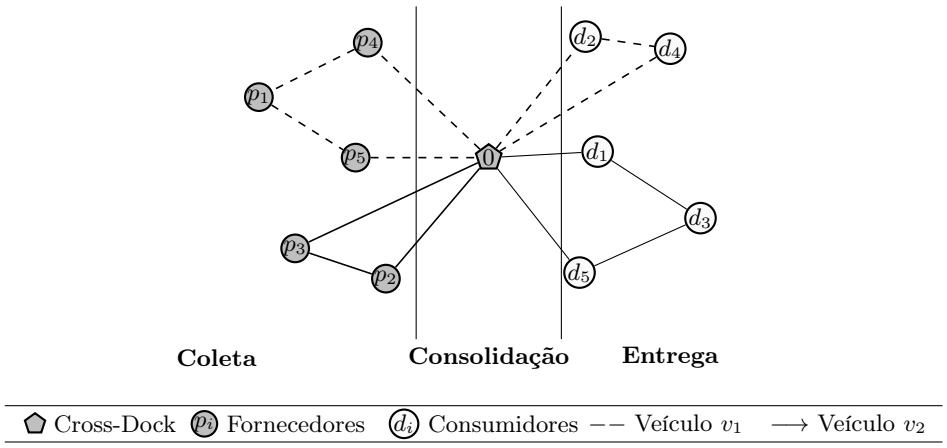

Figura 1.1: Ilustração de uma solução do VRPCD. São exibidas as rotas dos veículos v_1 e v_2. O primeiro parte do centro de consolidação, coleta os produtos das requisições r_4, r_1 e r_5 e retorna ao CD. O segundo veículo coleta os produtos de r_3 e r_2 e retorna ao CD. As requisições r_1, r_2 e r_5 são consolidadas no CD. Por fim, a mesma frota de veículos entrega os produtos nos seus respectivos consumidores.

O processo de consolidação das requisições r_1, r_2 e r_5 é detalhado na Figura 1.2.

No exemplo da Figura 1.1, os produtos das requisições r_1 e r_5, coletados pelo veículo v_1, e os produtos da requisição r_2, coletados pelo veículo v_2, são consolidados no CD. Em seguida, o veículo v_1 entrega os produtos das requisições r_2 e r_4, enquanto o veículo v_2 entrega os produtos das requisições r_1, r_3 e r_5.

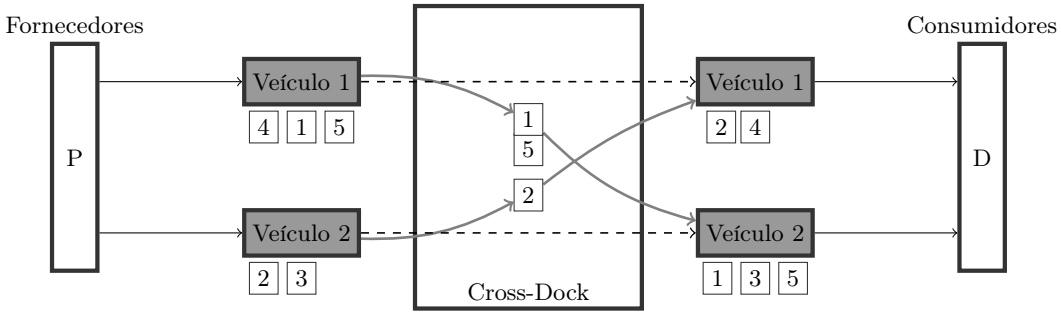

Figura 1.2: Exemplo do processo de consolidação. O veículo v_1 coleta os produtos das requisições r_4, r_1 e r_5 nos fornecedores, ao mesmo tempo em que o veículo v_2 coleta os produtos de r_2 e r_3. Em seguida, os veículos consolidam as requisições r_1, r_2 e r_5 no CD. Após o processo de consolidação o veículo v_1 entrega os produtos das requisições r_2 e r_4 e o veículo v_2 entrega os produtos das requisições r_1, r_3 e r_5 aos seus consumidores.

Dois cenários podem ser considerados no VRPCD. No primeiro, os veículos coletam e entregam os produtos de suas requisições sem realizar consolidação no CD. No segundo, os veículos realizam o processo de consolidação, ou seja, pelo menos uma de suas requisições tem os produtos coletados e entregues por veículos diferentes. O segundo cenário é composto por dois casos: (i) um veículo pode coletar os produtos de uma requisição no fornecedor mas não entregá-los ao seu consumidor; (ii) um veículo pode não coletar os produtos de uma requisição no seu fornecedor mas pode entregá-los ao seu consumidor.

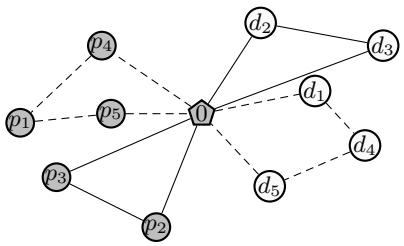

Fig. (a): Primeiro cenário. O veículo v_1 coleta e entrega os produtos das requisições r_1, r_4 e r_5, passando pelo CD sem realizar o processo de consolidação. O veículo v_2 coleta e entrega os produtos das requisições r_2 e r_3 sem realizar consolidação no CD.

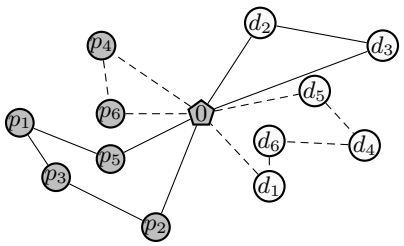

Fig. (b): Segundo cenário. O veículo v_1 coleta os produtos das requisições r_4 e r_6, retorna ao CD, onde realiza a consolidação e em seguida entrega os produtos das requisições r_1, r_4, r_5 e r_6. v_2 coleta os produtos das requisições r_1, r_2, r_3 e r_5, retorna ao CD, onde realiza a consolidação e em seguida entrega os produtos das requisições r_2 e r_3.

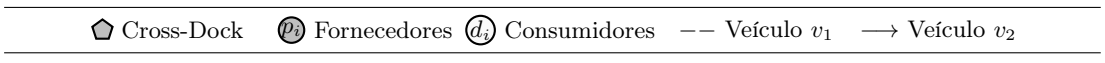

Figura 1.3: Ilustração dos dois possíveis cenários de consolidação do VRPCD.

A Figura 1.3(a) ilustra o primeiro cenário do VRPCD. O veículo v_1 coleta e entrega os produtos das requisições r_1, r_4 e r_5 e o veículo v_2 transporta os produtos das requisições r_2 e r_3, sem a necessidade de realizar consolidação. O segundo cenário do VRPCD é exemplificado na Figura 1.3(b), onde os produtos das requisições r_1, r_2, r_3 e r_5 são coletados pelo veículo v_2, enquanto os produtos das requisições r_4 e r_6 são coletados pelo v_1, no processo de entrega os produtos das requisições r_2 e r_3 são entregues por v_2 e os demais são entregues pelo veículo v_1. Neste exemplo, as requisições r_1 e r_5 são consolidadas no CD.

Aspectos práticos e teóricos motivam o estudo e a resolução de VRPCD. Em termos operacionais a implementação de soluções para problemas de distribuição podem representar ganhos consideráveis, uma vez que, segundo Alvarenga et al. [2007] de 10% a 15% do valor das mercadorias comercializadas no mundo advêm dos custos de transportes. Empresas como a rede de supermercados Wal Mart [Jayaraman & Ross, 2003], a rede varejista de farmácias Walgreens [Chen et al., 2006], a montadora de veículos Toyota [Witt, 1998] e empresa dinamarquesa de logística e consultoria Transvision [Wen et al., 2009], já empregam com sucesso técnicas de *Cross-Docking*. Em termos teóricos VRPCD pertence à classe de problemas NP-Difícil, uma vez que pode ser reduzido ao Problema de Roteamento de Veículos (VRP, do inglês *Vehicle Routing Problem*) clássico [Lee et al., 2006].

Como não são conhecidos algoritmos polinomiais para essa classe de problemas, a resolução do VRPCD para instâncias de grande porte por meio de métodos exatos pode exigir um grande tempo computacional. Fato este que motiva o desenvolvimento de heurísticas e até mesmo algoritmos híbridos baseados em programação matemática, que podem encontrar soluções próximas às soluções ótimas, com um baixo custo computacional [Ribas et al., 2010]. Nesta dissertação são propostas heurísticas eficientes, baseadas em metaheurísticas e procedimentos exatos para resolver uma versão de VRPCD.

As principais contribuições deste trabalho são: (i) o desenvolvimento de três heurísticas construtivas para VRPCD, baseadas em métodos clássicos da literatura de VRP; (ii) o desenvolvimento de uma heurística baseada na metaheurística *Iterated Local Search* (ILS), que trabalha apenas no espaço viável de soluções; (iii) desenvolvimento de uma matheurística baseada em ILS e em um modelo de Particionamento de Conjuntos, que obtém resultados superiores aos apresentados na literatura.

O restante deste texto é organizado da seguinte forma. O Capítulo 2 apresenta uma revisão bibliográfica sobre diferentes versões de VRP e uma revisão detalhada sobre trabalhos relacionados ao VRPCD. No Capítulo 3, as heurísticas desenvolvidas para resolver o problema são descritas. A matheurística proposta é detalhada no Ca-

pítulo 4. Os experimentos computacionais são descritos no Capítulo 5. Por fim, as considerações finais são apresentadas no Capítulo 6.

Capítulo 2

Trabalhos Relacionados

Neste capítulo é apresentada uma revisão bibliográfica sobre diferentes versões de VRP e uma revisão mais detalhada dos trabalhos relacionados ao VRPCD. A Seção 2.1 apresenta o Problema de Roteamento de Veículos Capacitado (CVRP, do inglês *Capacitaded Vehicle Routing Problem*). Os Problemas de Coleta e Entrega (PDP, do inglês *Pickup and Delivery Problem*s) são apresentados na Seção 2.2. Algumas referências para o Problema de Roteamento de Veículos com Janela de Tempo (VRPTW, do inglês *Vehicle Routing Problem with Time Windows*) são descritas na Seção 2.3. Por fim, a revisão bibliográfica para VRPCD é apresentada na Seção 2.4.

2.1 Problema de Roteamento de Veículos Capacitado

Seja G=(N,E) um grafo completo com o conjunto de nós $N = \{0\} \cup D$, em que o nó 0 representa o depósito, $D = \{1,\ldots,n\}$ um conjunto de consumidores e $E = \{(i,j) : i,j \in N\}$ o conjunto de arestas. Seja, ainda, $V = \{1, \ldots, k\}$ o conjunto de veículos homogêneos com capacidade limitada Q. Para cada consumidor $i \in D$ é associado uma demanda não negativa σ_i e para cada aresta $(i,j) \in E$ é associado um custo c_{ij}. CVRP consiste em definir rotas de custo mínimo para cada um dos veículos satisfazendo as seguintes restrições: (i) cada rota começa e termina no nó de depósito; (ii) cada consumidor deve ser visitado apenas uma vez e por um único veículo; (iii) o somatório das demandas de todos os consumidores de uma rota não deve exceder a capacidade do veículo.

Proposto por Dantzig & Ramser [1959] como uma generalização do Problema do Caixeiro Viajante (TSP, do inglês *Traveling Salesman Problem*) e portanto NP-Difícil,

CVRP foi a primeira versão de VRP a ser estudada. Baseadas neste diversas outras variantes foram propostas com um grande número de trabalhos publicados.

Na literatura são descritos procedimentos baseados tanto em abordagens exatas quanto heurísticas para resolver o CVRP. Laporte [1992], Gendreau et al. [2001] e Cordeau et al. [2005] apresentam heurísticas. Enquanto, Aragao & Uchoa [2003] e Fukasawa et al. [2006] apresentaram algoritmos exatos para o CVRP e provaram a otimalidade para um conjunto de instâncias por meio de um algoritmo *Branch-and-Cut-and-Price* (BCP) Robusto.

Os principais objetivos em se tratando de VRPs, definidos na literatura, são reduzir a distância total trafegada ou o número total de veículos utilizados. Em determinados contextos, quando uma demanda σ_i é maior que a capacidade do veículo, para algum consumidor $i \in D$, este consumidor deve ser visitado mais de uma vez, definindo o Problema de Roteamento de Veículos com Entregas Fracionadas (VRPSD, do inglês *Vehicle Routing with Split Delivery*), referenciado por Dror et al. [1994]. Caso a frota de veículos seja heterogênea, o problema em questão é o Problema de Roteamento de Veículos com Frota Heterogênea (HFVRP, do inglês *Heretogeneous Fleet Vehicle Routing Problem*), abordado em Penna et al. [2011]. Quando restrições quanto à distância máxima trafegada são impostas, define-se o Problema de Roteamento de Veículos com Restrição de Distância (DVRP, do inglês *Distance Constrained Vehicle Routing Problem*). Nagarajan & Ravi [2012] apresentam um algoritmo aproximativo para este problema.

2.2 Problemas de Coleta e Entrega

Seja $N = P \cup D$, onde $P = \{p_1, p_2, ..., p_m\}$ é um conjunto de nós de coleta e $D = \{d_1, d_2, ..., d_n\}$ é um conjunto de nós de entrega. O Problema de Coleta e Entrega (PDP, do inglês *Pickup and Delivery Problem*), consiste em definir rotas de coleta ou entrega satisfazendo as demandas de cada nó $i \in P \cup D$. Caso o processo de coleta seja realizado após a entrega o problema é referido como Problema de Roteamento de Veículos com *Backhaul* (VRPB, do inglês *Vehicle Routing Problem with Backhaul*) [Mosheiov, 1998].

PDP pode ser divido em dois grupos de problemas. (i) O primeiro, em que o transporte de produtos é realizado do depósito para os consumidores e dos consumidores para o depósito. (ii) O segundo, onde o transporte de cargas ocorre de consumidor a consumidor. O Problema de Roteamento de Veículos com Coleta e Entrega (VRPPD, do inglês *Vehicle Routing Problem with Pickup and Delivery*) é um caso do segundo

grupo [Parragh et al., 2008], em que o objetivo é reduzir a distância total trafegada pela frota de veículos. Um problema pertencente ao primeiro grupo é o Problema de Roteamento de Veículos com Coleta e Entrega Simultâneas (VRPSPD, do inglês *Vehicle Routing Problem with Simultaneous Pickups and Deliveries*) proposto por Min [1989], onde cada nó é associado a uma demanda de coleta e outra de entrega. Diversas variantes do PDP são detalhadas por Berbeglia et al. [2007] e Parragh et al. [2008].

Montané et al. [1997] definiram o Problema de Entrega Expressa (EDP, do inglês *Express Delivery Problem*), em que é preciso atender as demandas de um par ordenado origem-destino, utilizando um único ponto central, chamado de *hub*. O problema consiste em duas fases distintas: a fase de coleta, onde um veículo parte do *hub* atende os consumidores e retorna ao ponto de origem e a fase de entrega, onde é realizada a entrega a partir do ponto central.

Métodos heurísticos para VRPPD e VRPSPD podem ser encontrados nos trabalhos de Dethloff [2001], Assis [2007] e Subramanian [2012]. Este último ainda apresenta um *framework* baseado em *Brach-and-cut* para diversas variantes do VRP. Outro trabalho a apresentar procedimentos matemáticos para VRPPD e outras variantes do VRP é o de Salani [2005].

2.3 Problema de Roteamento de Veículos com Janela de Tempo

Seja G=(N,E) um grafo completo com o conjunto de nós N=$\{0\}\cup D$, em que o nó 0 representa o depósito, D=$\{1,\ldots,\ n\}$ um conjunto de consumidores e E=$\{(i,j)$: $i,j \in N\}$ um conjunto de arestas. Cada consumidor $i \in D$ deve ser visitado uma única vez dentro de um determinado intervalo de tempo $[a_i, b_i]$, ou seja, o veículo deve atender o consumidor i depois do tempo a_i e antes do tempo b_i. A cada aresta $(i,j) \in E$ é associado um custo c_{ij} e um tempo $t_{i,j}$ para se percorrer do consumidor i ao j. VRPTW consiste em determinar rotas de entrega de produtos do depósito central aos consumidores do conjunto D, respeitando as restrições de capacidade e janela de tempo. Duas das mais importantes versões de VRPTW definidas na literatura são avaliadas sob os seguintes objetivos: (i) definir rotas de custo mínimo ou duração mínima para uma frota de veículos homogêneos e (ii) definir um número mínimo de veículos para atender todas as demandas do consumidores.

Com o objetivo de definir rotas de custo mínimo, Ribas et al. [2010] apresentaram um algoritmo híbrido baseado nas metaheurísticas ILS e VND, além de uma formulação por Particionamento de Conjuntos. Alvarenga et al. [2007] e Brandão &

Vasconcelos [2010] seguiram o mesmo princípio e desenvolveram métodos híbridos, o primeiro numa abordagem inspirada em algoritmos genéticos e o segundo no algoritmo *Simulated Annealing* (SA). Balakrishnan [1993] apresentou heurísticas baseadas nas heurísticas vizinho mais próximo e *Savins* [Clarke & Wright, 1964] com a motivação de reduzir o número de veículos, Ombuki et al. [2006] também abordaram o problema com este objetivo.

Atualmente os melhores resultados para o VRPTW, com o objetivo de minimizar o custo total trafegado, foram os obtidos pelo algoritmo híbrido de Nagata et al. [2010]. Este método combina um eficiente operador de *crossover* com buscas locais extremamente eficientes. Os algoritmos ILS de Ibaraki et al. [2008], o *Adaptive Large Neighbourhood Search* (ALNS) de Pisinger & Ropke [2007], e o *Unified Tabu Search* (UTS) de Cordeau et al. [1997, 2001] também devem ser mencionados. Recentemente, Vidal et al. [2013] apresentaram uma detalhada revisão da literatura para o problema e algumas de suas variantes, além de propor um algoritmo híbrido para resolver estes problemas com um grande número de consumidores. Uma revisão mais ampla da literatura de VRPTW pode ser obtida nos trabalhos de Bräysy & Gendreau [2005a,b] e Golden et al. [2008].

2.4 Problema de Roteamento de Veículos com Cross-Docking

VRPCD é uma extensão de VRPPD que considera uma etapa de *Cross-Docking*. Na literatura, os trabalhos de Lee et al. [2006]; Liao et al. [2010]; Wen et al. [2009]; Tarantilis [2012] e Santos et al. [2010, 2011b,a] abordam as diferentes versões do problema. Estes consideraram algumas das seguintes restrições. (1) Cada rota, tanto de coleta quanto de entrega, iniciam e terminam no CD. (2) Cada nó $i \in P \cup D$ deve ser visitado uma única vez. (3) A mesma frota de veículos deve ser usada tanto no processo de coleta quanto no de entrega. (4) A capacidade dos veículos não deve ser extrapolada em nenhum ponto de uma rota. (5) Cada fornecedor (ou consumidor), assim como em VRPTW, deve ser visitado dentro de uma janela de tempo $[a_i, b_i]$, $\forall i \in P \cup D$. (6) Toda a etapa de roteamento deve ocorrer dentro de um horizonte de planejamento T; (7) Todos os veículos devem estar no CD durante a etapa de consolidação.

Lee et al. [2006] propuseram a primeira versão de VRPCD. Nesta versão, são consideradas todas as restrições citadas anteriormente (1 a 7). Além disso, a cada veículo é associado um custo operacional. O objetivo é minimizar os custos de transportes associados ao custo operacional dos veículos utilizados e aos custos de roteamento. Para

resolver o problema, os autores propuseram um algoritmo baseado na metaheurística *Tabu Search* (TS).

Liao et al. [2010] abordaram a mesma versão do problema proposta em Lee et al. [2006] e também propuseram um algoritmo baseado em Busca Tabu. As principais diferenças entre esses dois trabalhos são suas heurísticas para geração da solução inicial e as buscas locais. Em Liao et al. [2010] é utilizado o operador de busca local *Realocate*, que remove consumidores de um veículo e os atribuí a outro. Já o algoritmo de Lee et al. [2006] é baseado no operador *Exchange*, que troca consumidores entre pares de rotas distintas. Outra diferença é que apenas o algoritmo de Liao et al. [2010] admite decrementar o número de veículos utilizados durante a execução do algoritmo. Os experimentos foram realizados com o mesmo conjunto de instâncias e o algoritmo de Liao et al. [2010] obteve melhores resultados.

Em Wen et al. [2009] foi apresentada uma versão de VRPCD diferente daquela proposta por Lee et al. [2006]. Nesta versão os veículos passam pelo processo de consolidação tão logo terminem de realizar a coleta, sem a necessidade de estarem simultaneamente no CD. O processo de consolidação demanda um tempo para carga e descarga de produtos no CD, tempo este que é somado ao cálculo da janela de tempo na atividade de entrega. O objetivo do problema é definir um conjunto de rotas de coleta e entrega de tempo mínimo que passam obrigatoriamente pelo CD. Eles propuseram um conjunto de instâncias, uma formulação matemática para o problema e uma heurística eficiente baseada em TS.

Tarantilis [2012] abordou a mesma versão do problema definida em Wen et al. [2009]. Este autor propôs um algoritmo baseado em um procedimento adaptativo *multi-restart* associado à metaheurística TS, capaz de resolver as versões do problema com rotas abertas e rotas fechadas. A versão do problema com rotas abertas difere daquela com rotas fechadas em função das rotas iniciarem nos fornecedores e terminarem nos consumidores e não no CD. Tarantilis [2012] obteve melhores resultados que Wen et al. [2009], além de apresentar diferentes operadores de busca local.

Santos et al. [2010, 2011b,a] abordaram outra versão de VRPCD, onde é considerado o custo fixo em se manipular as mercadorias no centro de consolidação. Porém, o roteamento não é limitado pelas restrições de janela de tempo. Nesta variante, também é considerado que todos os veículos devem estar simultaneamente no CD durante a consolidação. As principais contribuições de Santos et al. [2010] foram a proposta de modelos matemáticos e de algoritmos exatos baseados nestas formulações. Em Santos et al. [2011b] e Santos et al. [2011a], são apresentadas uma formulação por geração de colunas e um algoritmo *branch-and-price* para o problema, respectivamente. Nestes trabalhos, soluções ótimas para instâncias com até 30 requisições foram obtidas.

Musa et al. [2010] direcionaram seus trabalhos ao problema de redes de *Cross-docking*, onde as cargas são transferidas de pontos de origem para pontos de destino passando por facilidades de *cross-docking*. Neste problema, busca-se minimizar o custo de transporte na rede, onde os veículos são carregados nos fornecedores e descarregados nos consumidores. O transporte pode ocorrer direta ou indiretamente por meio do CD. A principal diferença do trabalho de Musa et al. [2010] em relação ao VRPCD tratado nesta dissertação, e no trabalho de Wen et al. [2009], é que no contexto explorado por aqueles autores o objetivo do problema é criar um conjunto de rotas abertas que cobrem pares de pontos origem e destino, em que os veículos não iniciam e terminam suas rotas nos pontos centrais.

2.4.1 Formulação matemática

A versão de VRPCD estudada nesta dissertação foi proposta por Wen et al. [2009]. Estes apresentaram uma formulação matemática, em que o conjunto de nós é dado por $N{=}P \cup D \cup O$, onde $P{=}\{p_1,..., p_n\}$ é um conjunto de fornecedores, $D{=}\{d_1,..., d_n\}$ é um conjunto de consumidores e $O = \{o_1, o_2, o_3\ e\ o_4\}$ representa o CD. Os nós o_1 e $o_2 \in O$ representam os locais de início e término das rotas de coleta, enquanto o_3 e $o_4 \in O$ são os locais de início e término das rotas de entrega, respectivamente. $E{=}\{(i,j) \mid i,j \in P\} \cup \{(i,j) \mid i,j \in D\} \cup \{(0,j) \mid j \in \{P \cup D\}\}$ é um conjunto de arestas e $R = \{r_1,..., r_n\}$ um conjunto de requisições, em que cada requisição $r_i \in R$ é associada a um par $\{p_i, d_i\}$, tal que, $p_i \in P$ e $d_i \in D$. É considerado uma frota V de veículos homogêneos e os seguintes parâmetros: custo c_{ij} (em tempo) da aresta $(i,j) \in E$; janela de tempo $[a_i, b_i]$ do nó $i \in P \cup D$; demanda σ_i de uma requisição $r_i \in R$; a capacidade Q dos veículos; um tempo fixo A de preparo do veículo no CD; e um tempo B para carregar (ou descarregar) cada unidade de produto de uma requisição no CD.

As variáveis de decisão do modelo são: x_{ij}^k, que assume 1 se o veículo $k \in V$ percorre a aresta $(i,\ j)$ (0, caso contrário); u_i^k, que é igual a 1 se o veículo $k \in V$ descarrega a requisição i no CD (0, caso contrário); r_i^k, que indica se o veículo $k \in V$ recarrega a requisição i no CD; g_k, que diz se o veículo $k \in V$ é descarregado no CD; h_k, que indica se o veículo $k \in V$ é carregado no CD. Ainda são usadas as variáveis de tempo: s_i^k, o tempo em que o veículo $k \in V$ deixa o nó $i \in N$; t_k, o tempo no qual o veículo $k \in V$ termina de descarregar no CD; w_k, o tempo no qual o veículo $k \in V$ começa a recarregar no CD; e ν_i, o tempo no qual os produtos da requisição $i \in R$ devem ser coletados no CD.

Sendo assim, considerando que M é uma constante muito grande, a formulação

matemática de VRPCD é dada por:

$$Minimizar \sum_{(i,j)\in E} \sum_{k\in V} c_{ij} x_{ij}^k \tag{2.1}$$

Sujeito a:

$$\sum_{j:(i,j)\in E} \sum_{k\in V} x_{ij}^k = 1, \qquad \forall i \in P \cup D \tag{2.2}$$

$$\sum_{i\in P} \sum_{j:(i,j)\in E} \sigma_i x_{ij}^k \leq Q, \qquad \forall k \in V \tag{2.3}$$

$$\sum_{i\in D} \sum_{j:(i,j)\in E} \sigma_i x_{ij}^k \leq Q, \qquad \forall k \in V \tag{2.4}$$

$$\sum_{j:(h,j)\in E} x_{hj}^k = 1, \qquad \forall h \in \{o_1, o_3\}, k \in V \tag{2.5}$$

$$\sum_{j:(h,j)\in E} x_{jh}^k - \sum_{j:(h,j)\in E} x_{hj}^k = 0, \qquad \forall h \in P \cup D, k \in V \tag{2.6}$$

$$\sum_{j:(h,j)\in E} x_{jh}^k = 1, \qquad \forall h \in \{o_2, o_4\}, k \in V \tag{2.7}$$

$$s_j^k \geq s_i^k + c_{ij} - M \cdot (1 - x_{ij}^k), \qquad \forall (i,j) \in E, k \in V \tag{2.8}$$

$$a_i \leq s_i^k \leq b_i, \qquad \forall i \in N, k \in V \tag{2.9}$$

$$u_i^k - r_i^k = \sum_{j\in P\cup\{o_2\}} x_{ij}^k - \sum_{j\in D\cup\{o_4\}} x_{i+n,j}^k, \qquad \forall i \in P, k \in V \tag{2.10}$$

$$u_i^k + r_i^k \leq 1, \qquad \forall i \in P, k \in V \tag{2.11}$$

$$\frac{1}{M} \cdot \sum_{i\in P} u_i^k \leq g_k \leq \sum_{j\in P} u_i^k, \qquad \forall k \in V \tag{2.12}$$

$$t^k = s_{o2}^k + A \cdot g_k + B \cdot \sum_{i\in P} \sigma_i u_i^k, \qquad \forall k \in V \tag{2.13}$$

$$w_k \geq t_k, \qquad \forall k \in V \tag{2.14}$$

$$w_k \geq \nu_i - M \cdot (1 - r_i^k), \qquad \forall i \in P, k \in V \tag{2.15}$$

$$\nu_i \geq t_k - M \cdot (1 - u_i^k), \qquad \forall i \in P, k \in V \tag{2.16}$$

$$\frac{1}{M} \cdot \sum_{i \in P} r_i^k \leq h_k \leq \sum_{i \in P} r_i^k, \qquad \forall k \in V \tag{2.17}$$

$$s_{o3}^k = w_k + A \cdot h_k + B \cdot \sum_{i \in P} \sigma_i r_i^k, \qquad \forall k \in V \tag{2.18}$$

$$x_{ij}^k, u_i^k, r_i^k, g_k, h_k \in \{0, 1\}, \qquad \forall i \in P, (i, j) \in E, k \in V \tag{2.19}$$

$$s_i^k, t_k, w_k \geq 0, \qquad \forall i \in N, k \in V \tag{2.20}$$

$$\nu_i \geq 0, \qquad \forall i \in P. \tag{2.21}$$

A função objetivo (2.1) minimiza o tempo total gasto pelos veículos para percorrer seus trajetos. A restrição (2.2) assegura que cada nó seja visitado por exatamente um veículo, uma única vez. As restrições (2.3) e (2.4) garantem que a capacidade dos veículos, tanto na coleta quanto na entrega, não sejam extrapoladas. A restrição (2.5) restringe que apenas uma rota de coleta e uma de entrega sejam definidas, para cada veículo, a partir do CD. A restrição (2.6) garante a conservação de fluxo. A restrição (2.7) força que as rotas retornem ao CD depois da coleta e da entrega. A restrição (2.8) define o tempo para um veículo percorrer dois nós consecutivamente. Já a restrição (2.9) obriga que cada nó seja visitado dentro de sua janela de tempo e que toda a operação ocorra dentro do horizonte de planejamento.

As restrições de (2.10) a (2.18) exprimem a relação entre consumidores e fornecedores e tratam do processo de consolidação no CD. Mais precisamente, as restrições (2.10) e (2.11) expressam a ligação entre coleta e entrega. A restrição (2.12) define o valor de g_k. A restrição (2.13) impõe que o tempo que um veículo k termina de descarregar no CD deve ser igual a soma do (i) tempo que o veículo chega no CD, (ii) do tempo fixo de preparo do veículo e (iii) do tempo necessário para descarregar cada unidade de requisição transportada.

As restrições (2.14) a (2.16) definem a ordem das atividades no CD. Cada veículo só deve começar a carregar depois de realizar a descarga de produtos no CD. Os produtos de uma requisição r_i só podem ser recarregados por um veículo depois que estes forem descarregados por outro (2.16). Consequentemente, o tempo de inicio para recarregar os produtos da requisição r_i deve ser posterior ao tempo dos mesmos terem sido descarregados no CD. As restrições (2.17) e (2.18) descrevem o tempo no qual o veículo está preparado para deixar o CD e partir em direção aos consumidores. Por fim, as restrições (2.19) a (2.21) definem o domínio das variáveis de decisão do modelo.

Como pode ser observado, todos os trabalhos que tratam heuristicamente de

VRPCD propuseram algoritmos baseados em Busca Tatu. No entanto, nesta dissertação uma técnica diferente é utilizada. São propostas heurísticas baseadas na metaheurística ILS. Estas trabalham apenas no espaço viável de soluções e são capazes obter soluções comparáveis às da literatura.

Capítulo 3

Heurísticas para VRPCD

VRPCD é uma extensão do clássico VRP, portanto ele é NP-Difícil [Lee et al., 2006]. Diversos algoritmos propostos para este último podem ser estendidos para resolver o VRPCD. Neste Capítulo, são apresentadas as heurísticas desenvolvidas para resolver o problema. Na Seção 3.1 são apresentadas as heurísticas construtivas propostas. As buscas locais e os mecanismos de perturbação são detalhados na Seção 3.2. Por fim, a metaheurística ILS é apresentada na Seção 3.3.

3.1 Heurísticas Construtivas

Heurísticas construtivas são algoritmos que geram uma solução viável para um determinado problema com base em um conjunto de regras de construção. O principal objetivo destes procedimentos é construir soluções de boa qualidade com baixo custo computacional. Neste trabalho, são apresentadas três heurísticas para a geração de uma solução para VRPCD: *Two-Side Nearest Insertion* (2S-NI), *Two-Side Nearest Neighbor* (2S-NN) e *Two-Side Savings* (2S-S). Estas heurísticas não consideram o processo de consolidação, uma vez que nas soluções por elas geradas, o veículo que coleta uma requisição no fornecedor é sempre o mesmo que a entrega ao consumidor.

3.1.1 Two-Side Nearest Insertion

2S-NI é heurística construtiva baseada no algoritmo *Nearest Insertion* (NI), proposto por Rosenkrantz et al. [1977] para TSP. Seja o Conjunto de Requisições Candidatas (CRR), que inicialmente contém todas as requisições de R, $c^p_{i,j}$ o custo da aresta (p_i, p_j) e $c^d_{i,j}$ o custo da aresta (d_i, d_j), onde p_i e d_i são os respectivos nós fornecedor e consumidor de uma requisição r_i já atribuída a um veículo, e p_j e d_j os respectivos nós

fornecedor e consumidor de uma requisição $r_j \in$ CRR. 2S-NI atribui iterativamente cada uma das requisições de CRR a um veículo. O algoritmo, primeiramente, cria um par de rotas de coleta e entrega vazias (rotas que começam e terminam no CD, sem visitar nenhum outro ponto) para um veículo $v_k \in V$ e, a cada iteração atribui uma requisição $r_j \in$ CRR, com o menor valor de $\gamma_{i,j} = c^d_{i,j} + c^p_{i,j}$ (ou seja, a requisição que possuir fornecedores e consumidores mais próximos das rotas de v_k), ao veículo v_k. Quando a capacidade de um veículo é extrapolada um novo é criado. A atribuição de uma requisição $r_j \in$ CRR ao veículo v_k se dá pela inserção do fornecedor $p_j \in P$ e do consumidor $d_j \in D$ nas rotas de coleta e entrega de v_k, respectivamente. Esta inserção é feita na posição onde o acréscimo no custo da rota seja mínimo, respeitando as restrições do problema.

A Figura 3.1 ilustra uma iteração da heurística 2S-NI. Na Figura 3.1(a) uma solução parcial é apresentada, onde as requisições r_1, r_2 e r_3 já foram atribuídas ao veículo corrente v_k. Conforme ilustrado em 3.1(b), para cada requisição $r_j \in$ CRR é calculado o valor de $\gamma_{i,j}$. Na Figura 3.1(c), a requisição r_4 é atribuída ao veículo v_k, uma vez que esta possui o menor valor de $\gamma_{i,j}$.

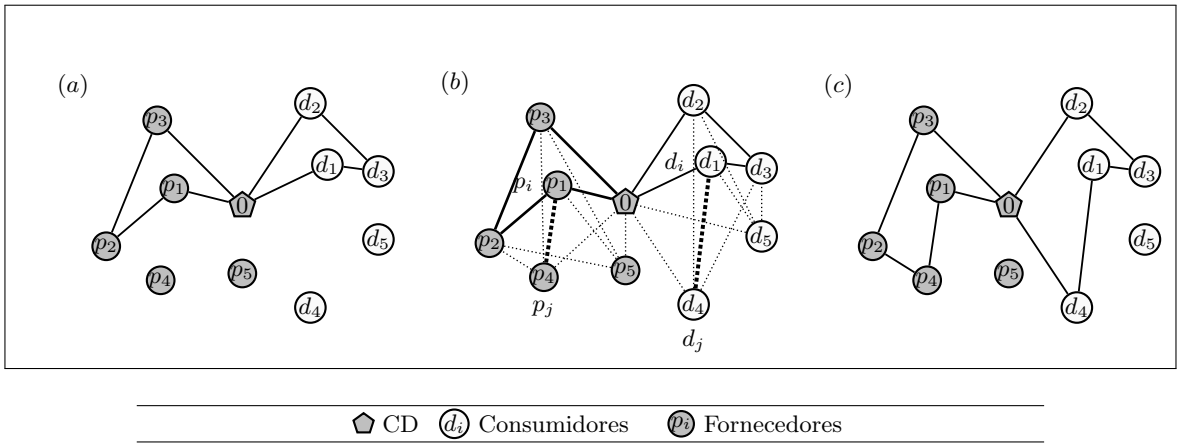

Figura 3.1: Ilustração de uma iteração da heurística 2S-NI. (a) mostra o estado da solução em que as requisições r_1, r_2 e r_3 já foram atribuídas ao veículo corrente. O próximo passo do algoritmo, demonstrado em (b), é definir a próxima requisição ainda não atribuída a ser alocada ao veículo. Na parte (c), a requisição r_4 é atribuída ao veículo uma vez que esta possui o menor valor de $\gamma_{i,j}$.

O pseudocódigo da 2S-NI é apresentado no Algoritmo 2. Nas linhas 1, 2 e 3, são inicializados, respectivamente, o conjunto de requisição CRR, com todas as requisições do conjunto R, uma solução vazia (sem nenhuma rota) S e o identificador do veículo v_k. Em seguida, na linha 4, uma requisição $r_j \in$ CRR é escolhida de forma aleatória. Nas linhas 5 e 6, são criadas duas rotas, uma de coleta (RC) e outra de entrega (RE), respectivamente. No laço das linhas 7–17, enquanto existir requisições em CRR, atribui

requisição por requisição ao veículo corrente v_k. Na linha 9 e 10, se a atribuição da requisição r_j não extrapola a capacidade do veículo v_k, o fornecedor p_j e o consumidor d_j são inseridos respectivamente em suas rotas de coleta e entrega, com complexidade no pior caso de $\mathcal{O}(|R|)$. Na linha 11, após atribuir r_j nas rotas do veículo corrente, remove a requisição do conjunto CRR. Caso a atribuição de r_j extrapole a capacidade do veículo, as rotas RC e RE são inseridas na solução S, na linha 13. Em seguida, novas rotas de coleta e entrega são criadas nas linhas 14 e 15. Um novo veículo é criado na linha 16. Após atribuir a requisição a um veículo, outra requisição r_j que possuir o menor valor de $\gamma_{i,j}$, é selecionada do conjunto CRR em $\mathcal{O}(|R|^2)$, no passo da linha 17. Ao final do procedimento, na linha 18, a solução S é retornada. A complexidade assintótica desta heurística é $\mathcal{O}(|R|^3)$.

Algoritmo 1: Heurística *Two-Side Nearest Insertion*

```
1  Inicialize CRR;
2  S ← ∅;
3  v_k ← 1;
4  Selecione uma requisição aleatória r_j de CRR;
5  RC ← Crie rota de coleta vazia;
6  RE ← Crie rota de entrega vazia;
7  enquanto existir requisições em CRR faça
8      se a atribuição de r_j não extrapola a capacidade do veículo v_k então
9          Insira p_j de r_j em RC;
10         Insira d_j de r_j em RE;
11         Remova r_j de CRR;
12     senão
13         Insira as rotas RC e RE em S;
14         RC ← Crie rota de coleta vazia;
15         RE ← Crie rota de entrega vazia;
16         v_k ← v_k+1;
17     Selecione a requisição r_j de CRR com o menor valor de γ_{i,j};
18 retorna S;
```

3.1.2 Two-Side Nearest Neighbor

2S-NN é uma heurística proposta para VRPCD baseada no algoritmo *Nearest-Neighbor*, proposto por Solomon [1987] para VRPTW. O algoritmo 2S-NN, primeiramente, cria um par de rotas de coleta e entrega vazias (rotas que começam e terminam no CD, sem visitar nenhum outro ponto) para um veículo $v_k \in V$. A cada iteração esta heurística atribui uma requisição $r_i \in$ CRR às rotas do veículo v_k até que todas as requisições do

conjunto CRR sejam atribuídas a um veículo. Quando a capacidade de um veículo é extrapolada um novo é criado. A atribuição de uma requisição $r_i \in$ CRR ao veículo v_k se dá pela inserção do fornecedor $p_i \in P$ e do consumidor $d_i \in D$ nas rotas de coleta e entrega de v_k, respectivamente.

Seja $c^p_{i,l}$ o custo da aresta $(p_i,\ p_l)$ e $c^d_{i,l}$ o custo da aresta $(d_i,\ d_l)$, onde p_l e d_l são os respectivos nós fornecedor e consumidor da última requisição $r_l \in R \backslash CRR$ já atribuída a um veículo e, p_i e d_i os nós fornecedor e consumidor de uma requisição $i \in$ CRR. Na heurística 2S-NN as requisições são atribuídas ao veículo sequencialmente ordenadas de acordo com o valor de $\gamma_{i,l} = c^d_{i,l} + c^p_{i,l}$. Neste método, os fornecedores e consumidores são inseridos nas respectivas rotas de coleta e entrega depois do último nó visitado.

A Figura 3.2 ilustra uma iteração da heurística 2S-NN. Na Figura 3.2(a) uma solução parcial é considerada, onde a requisição r_1 foi a última atribuída ao veículo na iteração anterior. Na iteração ilustrada, o valor de $\gamma_{i,1}$ é calculado entre a requisição r_1 e cada uma das requisições $r_i \in$ CRR. Em 3.2(b), a requisição com menor valor de $\gamma_{i,1}$ é atribuída às rotas do veículo corrente que, conforme ilustrado, é a requisição r_3.

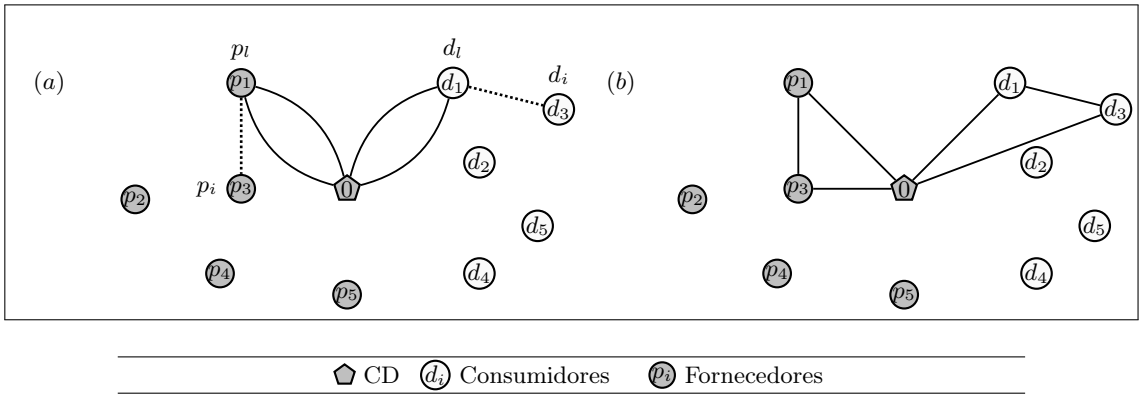

Figura 3.2: Ilustração de uma iteração da heurística 2S-NN. Em (a), assumindo-se que r_1 foi a última requisição atribuída ao veículo na iteração anterior, é calculado a requisição de CRR com o menor $\gamma_{i,l}$ em relação a r_l. Na Figura (b), a requisição r_3, definida como a de menor valor de $\gamma_{i,l}$, é atribuída ao veículo corrente inserindo os seus nós fornecedor e consumidor nas rotas de coleta e entrega, respectivamente.

O pseudocódigo de 2S-NN é apresentado no Algoritmo 2. Nas linhas 1, 2 e 3, são inicializados, respectivamente, o conjunto de requisição CRR, com todas as requisições do conjunto R, uma solução vazia (sem nenhuma rota) S e o identificador do veículo v_k. Em seguida, na linha 4, uma requisição $r_i \in$ CRR é escolhida de forma aleatória. Nas linhas 5 e 6, respectivamente, são criadas duas rotas, uma de coleta (RC) e outra de entrega (RE). No laço das linhas 7–18, enquanto existir requisições em CRR, atribui requisição por requisição à solução. A última requisição atribuída a um veículo, r_l, é atualizada na linha 9. Na linha 10, o fornecedor p_i da requisição r_i é inserido na rota

de coleta. Na linha 11, o consumidor d_i da requisição r_i é inserido na rota de entrega. Na linha 12, a requisição r_i é removida de CRR. Caso a atribuição de r_i extrapole a capacidade do veículo, as rotas RC e RE são inseridas na solução S, na linha 14. Em seguida, novas rotas de coleta e entrega são criadas nas linhas 15 e 16. Um novo veículo é criado na linha 17. Na linha 18, uma nova requisição r_i com o menor valor de $\gamma_{i,l}$ é selecionada de CRR, em $\mathcal{O}(|R|^2)$. Por fim, a solução definida é retornada na linha 18. A complexidade assintótica da heurística 2S-NN é $\mathcal{O}(|R|^3)$.

Algoritmo 2: Heurística *Two-Side Nearest Neighbor*

```
1  Inicialize CRR;
2  S ← ∅;
3  v_k ← 1;
4  Selecione uma requisição aleatória r_i de CRR;
5  RC ← Crie rota de coleta vazia;
6  RE ← Crie rota de entrega vazia;
7  enquanto existir requisições em CRR faça
8      se a atribuição de r_i não extrapola a capacidade do veículo então
9          r_l ← r_i;
10         Insira p_i de r_i em RC;
11         Insira d_i de r_i em RE;
12         Remova r_i de CRR;
13     senão
14         Insira as rotas RC e RE em S;
15         RC ← Crie rota de coleta vazia;
16         RE ← Crie rota de entrega vazia;
17         v_k ← v_k+1;
18     Selecione a requisição r_i de CRR com o menor valor de γ_{i,l} em relação a r_l;
19 retorna S;
```

3.1.3 Two-Side Savings

2S-S é uma heurística construtiva baseada no algoritmo *Savings*, proposto por Clarke & Wright [1964] para VRP. O algoritmo inicia com $|R|$ pares de rotas distintas, uma para cada $r_i \in R$, em que para cada veículo são definidas uma rota de coleta e outra de entrega. Em seguida, pares de rotas são iterativamente agrupadas numa tentativa de minimizar os custos de transporte.

Sejam $c^p_{i,j}$ o custo da aresta $(p_i,\ p_j)$ e $c^d_{i,j}$ o custo da aresta $(d_i,\ d_j)$. Seja ainda, duas requisições r_i e r_j atribuídas às rotas dos veículos v_i e v_j é definida a função de $sav(r_i,\ r_j) = (c^p_{i0} + c^p_{j0} - c^p_{ij}) + (c^d_{i0} + c^d_{j0} - c^d_{ij})$, onde $sav(r_i,\ r_j)$ é a distância economizada

que resulta do transporte das requisições r_i e r_j por um mesmo veículo ao contrário do transporte destas individualmente via CD. As rotas dos veículos v_i e v_j são agrupadas caso o valor de *Savings* minimize o custo objetivo total. Para que as rodas dos dois veículos sejam agrupadas, os fornecedores da rota de coleta de um veículo são inseridos na rota de coleta do outro veículo, o mesmo ocorre para as rotas de entrega. Esta inserção é feita na posição onde o acréscimo no custo da rota seja mínimo, respeitando as restrições do problema.

A Figura 3.3 ilustra uma iteração da heurística 2S-S. A Figura 3.3(a) apresenta cinco requisições e seus respectivos nós fornecedores e consumidores. Inicialmente, o procedimento atribui cada requisição a um veículo, definindo suas rotas de coleta e entrega, conforme exemplificado na Figura 3.3(b). Em seguida, na Figura 3.3(c), pares de rotas são agrupadas e atribuídas a um mesmo veículo, o que no exemplo é exemplificado pelas rotas das requisições r_3 e r_1.

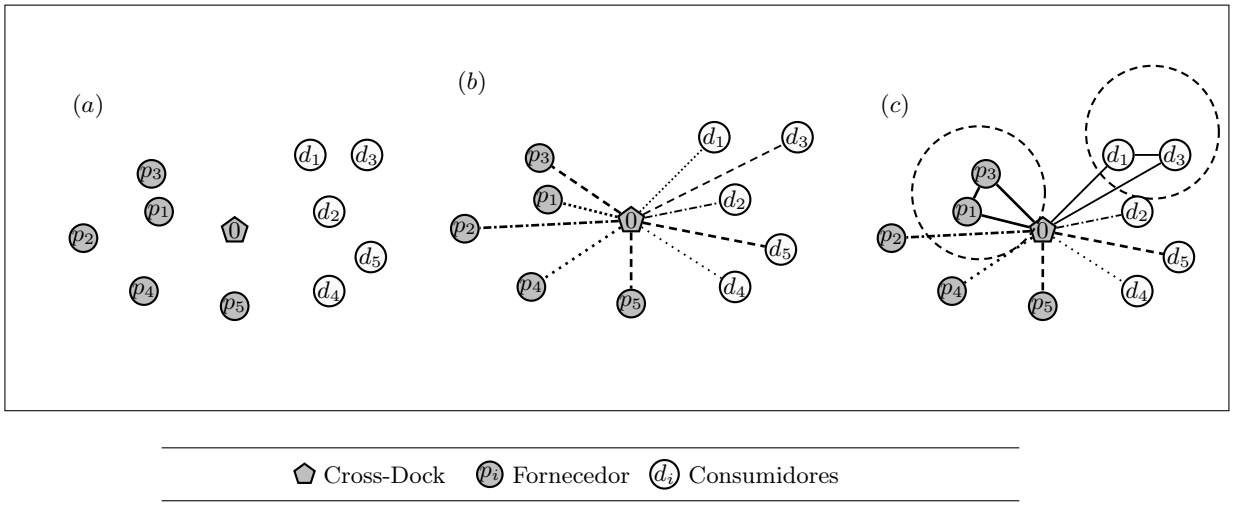

Figura 3.3: Ilustração de uma iteração da heurística 2S-S. Em (a), cinco requisições são distribuídas entre seus respectivos fornecedores e consumidores. Em seguida, cada uma das requisições é atribuída a um veículo, conforme ilustrado em (b). Na parte (c), o par de requisições r_3 e r_1 é associado ao mesmo veículo.

O pseudocódigo de 2S-S é apresentado no Algoritmo 3. Na linha 1, é definida uma solução inicial S em $\mathcal{O}(|R|)$, onde cada fornecedor de uma requisição é atribuído a uma rota de coleta e cada consumidor de uma requisição e associado a uma rota de entrega. No laço das linhas 2–14, as rotas são agrupadas aos pares com complexidade de $\mathcal{O}(|R|^2)$. Para cada par r_i e r_j, na linha 7, é calculado o valor de *savings* e armazenado o melhor par a ser agrupado (linhas 5–10). Na linha 12 o agrupamento é avaliado. Caso a nova rota seja viável, a solução S é atualizada na linha 13, com complexidade de $\mathcal{O}(|R|^2)$. Por fim, a solução S é retorna na linha 15. A heurística 2S-S possui complexidade assintótica de $\mathcal{O}(|R|^3)$.

Algoritmo 3: Heurística *Two-Side Savings*

```
1  Defina uma solução S, em que cada requisição é atribuída a um veículo;
2  para toda r_i ∈ R faça
3      r_a ← ∅;
4      cost^s ← ∞;
5      para toda r_j ∈ R faça
6          se (r_i ≠ r_j) e (r_i e r_j estão em rotas diferentes) então
7              Calcule o valor de sav(i, j) para as requisições r_i e r_j;
8              se sav(i, j) ≤ cost^s então
9                  cost^s ← sav(i, j);
10                 r_a ← rj;
11     se r_a ≠ ∅ então
12         Verifique a viabilidade em se agrupar as rotas das requisições r_i e r_a;
13         Agrupe as rotas de r_i e r_a na solução S;
14         cost^s ← ∞;
15 retorna S;
```

3.2 Buscas Locais e Perturbação

Neste trabalho, sete buscas locais e dois mecanismos de perturbação para VRPCD são apresentados. Dentre as buscas locais, duas realizam movimentos que não consideram o processo de consolidação, enquanto cinco delas consideram o processo de consolidação no CD. As buscas locais e perturbações são descritas a seguir.

3.2.1 Buscas Locais Interveiculares

Uma busca local interveicular avalia soluções que não consideram o processo de consolidação. Nestas, uma requisição é movida de um veículo a outro, para isto são atualizadas tanto as rotas de coleta quanto as de entrega. As duas buscas locais que realizam movimentos interveiculares são a *Exchange* e a *Realocate*.

A busca local *exchange* gera um conjunto de soluções vizinhas de uma solução s pela troca de duas requisições de veículos diferentes. Para cada par de requisições r_i e r_j, associadas a dois veículos v_i e v_j, a busca de local *exchange* avalia, em cada iteração, o custo global para mover r_i do veículo v_i para v_j e r_j de v_j para v_i. A primeira troca que resultar em uma solução melhor que s é efetivadas. A busca local *exchange* percorre uma vizinhança de tamanho $\mathcal{O}(|R|^2)$.

A busca local *realocate* define, para uma solução s, uma vizinhança composta por um conjunto de soluções gerado pela realoção de uma requisição entre pares de

veículos de s. A busca local *realocate* avalia, em cada iteração, o custo de se realocar uma requisição r_i de um veículo v_i para outro veículo v_j. Se o custo da nova solução for melhor que s esta é atualizada. A vizinhança *realocate* possui o tamanho de $\mathcal{O}(|R|^2)$.

A Figura 3.4 ilustra dois vizinhos definidos pelas buscas locais *exchange* e *realocate*. Assumindo-se a solução inicial, apresentada em 3.4(a). A Figura 3.4(b) ilustra um vizinho em *exchange*, onde a requisição r_6 do veículo v_1 é transferida para o veículo v_2 e a requisição r_3 é movida do veículo v_2 para o veículo v_1. Em 3.4(c) um vizinho é avaliado na busca local *realocate*, onde a requisição r_4 é movida do veículo v_3 para o veículo v_2.

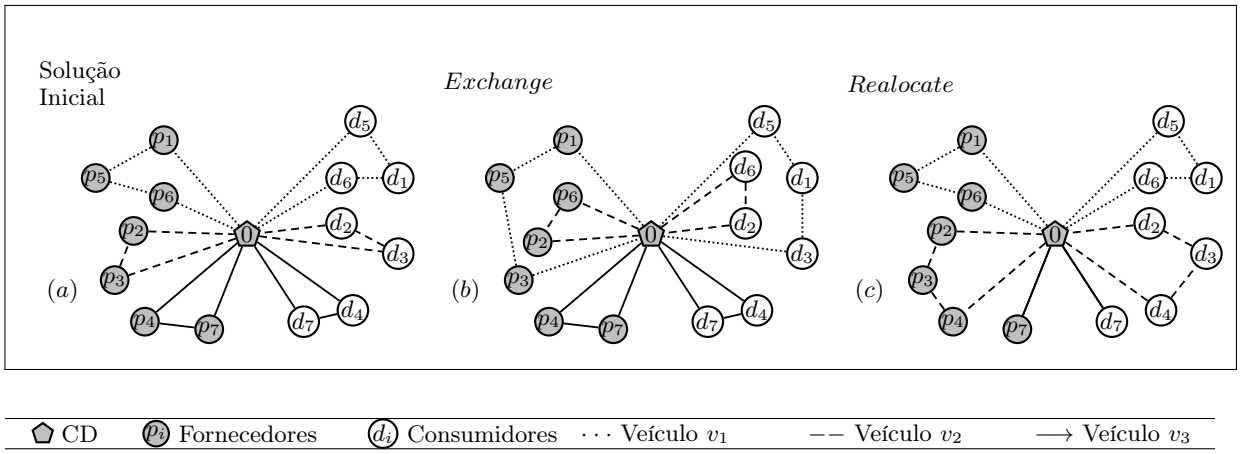

Figura 3.4: Buscas Locais Interveiculares. Dada uma solução inicial (a), são ilustrados dois vizinhos gerados pelas buscas locais *Exchange* (b) e *Realocate* (c), aplicadas ao primeiro cenário do VRPCD.

A busca local *Exchange*, foi implementada com a estratégia melhor aprimorante, ou seja, para cada troca entre pares de requisições são verificadas todas as possibilidades e efetuada a de menor custo. Enquanto a *realocate*, foi implementada sob a estratégia primeiro aprimorante, que efetiva o primeiro movimento aprimorante realizado.

3.2.2 Buscas Locais Inter-rotas

Das cinco buscas locais para o segundo cenário do problema, quatro avaliam soluções geradas por movimentos inter-rotas. Nestas um fornecedor (ou consumidor) de uma requisição é movido de uma rota de coleta (ou entrega) para outra rota de coleta (ou entrega) sem que para isto seja necessário realizar movimentos semelhantes nas rotas de entrega (ou coleta). São apresentadas quatro buscas locais inter-rotas: *Insertion*, *Swap*(2,1), *Drop route* e *Swap*(1,1), todas ilustradas na Figura 3.5. As três primeiras implementam movimentos primeiro aprimorantes, enquanto a última foi implementada

com a estratégia melhor aprimorante. O tamanho das vizinhanças geradas por estas buscas locais é $\mathcal{O}(|N|^2)$.

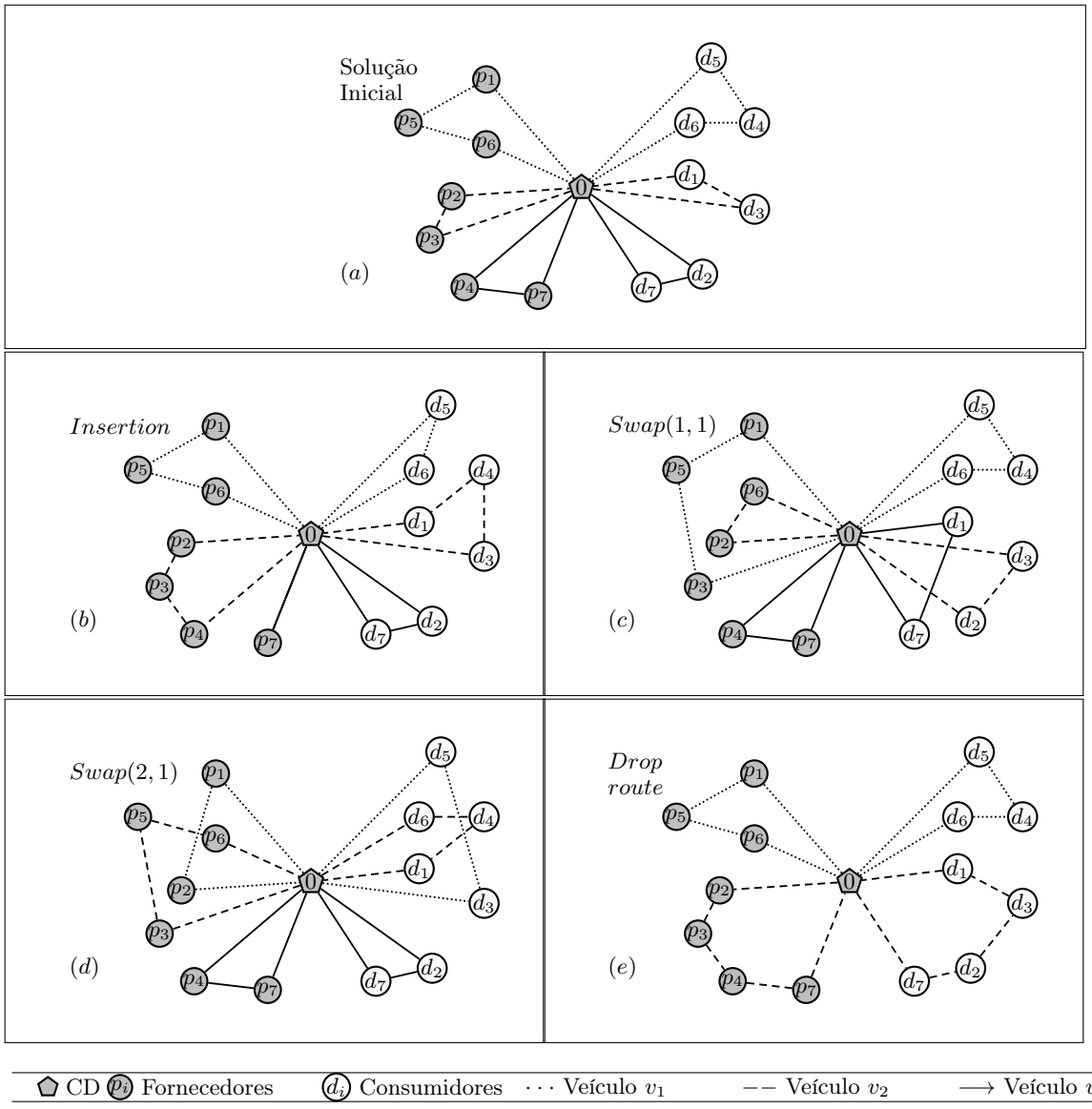

Figura 3.5: Buscas Locais Inter–rotas. Dada uma solução inicial (a), são ilustrados vizinhos gerados por movimentos executados pelas buscas locais *Insertion*(b), *Swap*(1,1)(c), *Swap*(2,1)(d) e *Drop route* (e), aplicados ao segundo cenário do VRPCD.

A busca local *Insertion* gera um conjunto de soluções vizinhas de uma solução s pela troca de um fornecedor (ou consumidor) em uma rota de coleta (ou entrega). Seja uma requisição r_j e seus respectivos nós de coleta $d_j \in P$ e entrega $d_j \in D$, estes são movidos de seus veículos de origem v_i para os veículos de destino v_j, nas rotas de coleta ou entrega. A Figura 3.5(b) ilustra um vizinho gerado pela busca local *Insertion*. No exemplo a requisição perturbada é r_4. Entre as rotas de coleta, a requisição r_4 é movida

do veículo v_3 para o v_2, ao mesmo tempo em que nas rotas de entrega a requisição r_4 é movida do veículo v_1 para o v_2.

Um conjunto de soluções vizinhas de uma solução s geradas pela busca local *Swap*(1,1) diferem de s pela perturbação entre um nó $i \in P \cup D$ de uma rota c_i e um nó $j \in P \cup D$ da rota c_j. Um vizinho gerado por esta busca local é ilustrado na Figura 3.5(c), onde nas rotas de coleta, a requisição r_3, coletada pelo veículo v_2, é perturbada com a requisição r_6 do veículo v_1, enquanto nas rotas de entrega, a requisição r_1, que seria entregue pelo veículo v_2, é trocada com a requisição r_2 do veículo v_3.

Swap(2,1) é uma busca local que gera o conjunto de soluções vizinhas de s onde dois nós adjacentes, i e j, são movidos da rota de um veículo v_i para a rota do veículo v_j, enquanto um nó t da rota de v_j é movido para a rota de v_i. Um vizinho gerado por este movimento é ilustrado na Figura 3.5(d), onde as requisições r_5 e r_6 são movidas da rota de coleta do veículo v_1 para a rota de coleta do veículo v_2, enquanto a requisição r_2 do veículo v_2 é movida para a rota de coleta do veículo v_1, ao mesmo tempo que nas rotas de entrega as requisições r_4 e r_6 são movidas de v_1 para v_2 e a requisição r_3 é movida de v_2 para v_1.

A busca local *drop route* gera um conjunto de soluções vizinhas de s pela remoção de uma de suas rota de coleta (ou entrega) e a atribuição de seus nós a outros veículos. Esta busca local realiza movimentos na tentativa de eliminar o número de rotas e consequentemente o número de veículos utilizados. A Figura 3.5(e) ilustra um vizinho gerado por esta busca local, onde as rotas do veículo v_3 são excluídas e as requisições destas são atribuídas a outros veículos. No exemplo ilustrado, observe que as requisições r_4 e r_7 nas rotas de coleta e as requisições r_2 e r_7 das rotas de entrega foram atribuídas ao veículo v_2.

3.2.3 Buscas Locais Intrarota

A vizinhança *ReInsertion Intraroute* de uma solução s é definida por um conjunto de soluções que diferem de s pelo posicionamento de um nó em uma rota de coleta ou entrega. Para cada veículo, uma requisição r_i é removida de uma rota de coleta (ou entrega) e inserida em outra posição da mesma rota. Esta vizinhança possui tamanho de $\mathcal{O}(|N|^2)$. A busca local que avalia esta vizinhança foi implementada com a estratégia melhor aprimorante. A Figura 3.6 ilustra uma iteração do procedimento, onde a requisição r_4 é movida de posição na rota de coleta do veículo v_2.

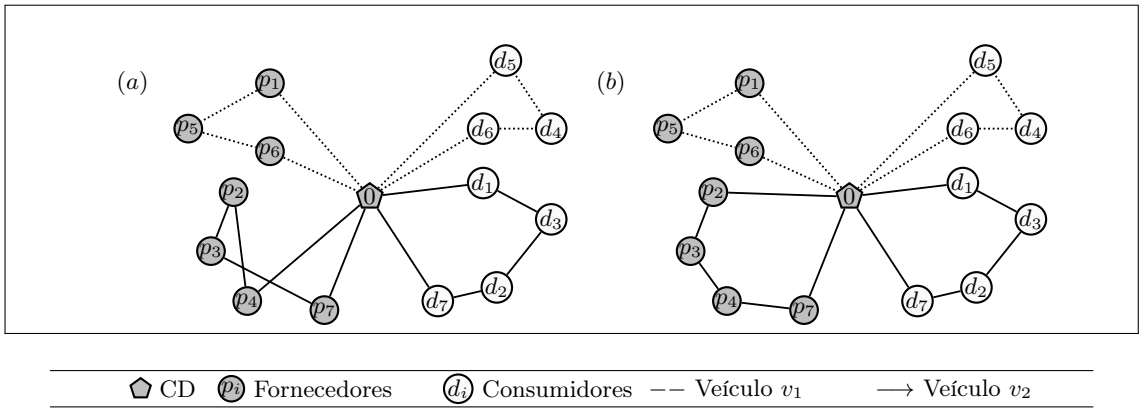

Figura 3.6: Buscas Locais Intrarota. Assumindo-se a solução inicial apresentada em (a), é ilustrado um movimento realizado pela busca local que avalia a vizinhança *ReInsertion Intraroute* (b).

3.2.4 Perturbações

Uma alternativa para melhorar o processo de busca ao alcançar um ótimo local é repetir a busca de outro ponto de partida [Lourenço et al., 2010]. Este princípio pode ser explorado por meio de uma perturbação em uma determinada solução. O intuito é possibilitar uma maior diversificação da busca sob o espaço de soluções. Para este trabalho duas perturbações são consideradas, a *Split* e a *Random–Exchange*.

A perturbação *Split* consiste em selecionar aleatoriamente uma solução s' na vizinhança *Split*. A vizinhança *Split* de uma solução s é definida pelo conjunto de soluções dado pela divisão das rotas de s em rotas menores. Um vizinho gerado por esta perturbação é ilustrado na Figura 3.7, onde as rotas do veículo v_2 são divididas em duas rotas menores. A complexidade assintótica no pior caso para gerar uma solução perturbada em *Split* é $\mathcal{O}(|N|^2)$.

A perturbação *Random-Exchange* consiste em selecionar aleatoriamente uma solução na vizinhança de *exchange*. Nesta, o procedimento de busca realiza uma sequência aleatória de movimentos, onde pares de requisições são trocadas entres dois veículos. Um vizinho gerado por esta perturbação é ilustrado na Figura 3.7, onde os pares de requisições r_4 e r_6 nas rotas de coleta e as requisições r_1 e r_4 nas rotas de entrega são permutados. A complexidade assintótica no pior caso para gerar uma solução perturbada em *Random–Exchange* é $\mathcal{O}(|N|^2)$.

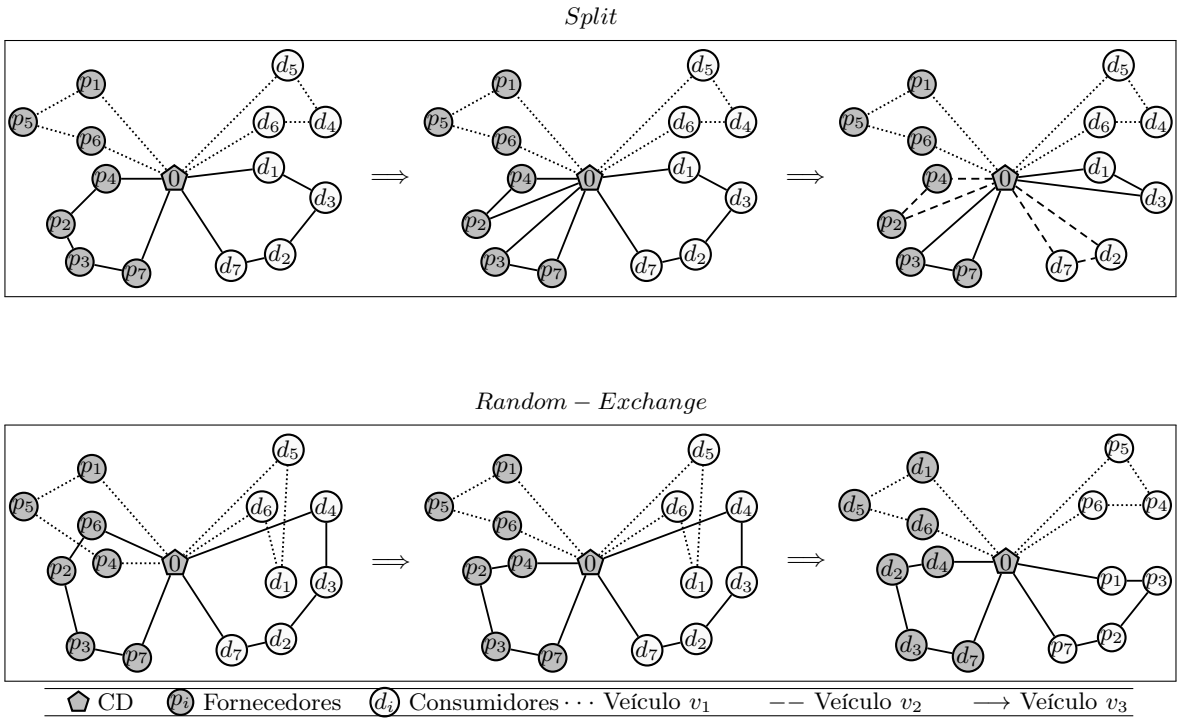

Figura 3.7: Ilustração de dois vizinhos gerados pelos operadores de perturbação *Split* e *Random–Exchange*. Dada uma solução inicial, em um movimento *Split* divide-se as rotas do veículo v_2 em duas rotas menores. Enquanto no *Random-Exchange*, as requisições r_4 e r_2 nas rodas de coleta e as requisições r_1 e r_4 nas rotas de entrega são trocadas.

3.3 Heurística ILS-VRPCD

Iterated Local Search (ILS) é uma metaheurística baseada em busca local. Um algoritmo baseado em ILS parte de uma solução viável e aplica sucessivamente busca local e perturbação, limitando o espaço de busca a um conjunto de ótimos locais [Lourenço et al., 2010]. As quatro fases de uma heurística baseada em ILS são: (i) *geração de solução inicial*, onde uma solução de partida é definida; (ii) *busca local*, responsável por examinar a vizinhança de determinada solução na procura de uma solução melhorante; (iii) *perturbação*, que possibilita gerar uma solução aleatória e diversificar a busca sob o espaço de soluções e (iv) *critério de aceitação*, que dado uma solução corrente s e um vizinho s', diz se a solução s é atualizada por s' dado um fator de aceitação $\alpha\%$.

Variable Neighborhood Descent (VND) é uma metaheurística, que consiste na combinação de um conjunto de buscas locais, baseada no fato de que um mínimo local em uma vizinhança não ser o mínimo local de outra [Gendreau & Potvin, 2010]. Assim, VND percorre o espaço de busca, por meio de trocas sistemáticas entre as buscas locais a cada vez que um novo ótimo local é definido. O objetivo desta é alcança uma solução que é um mínimo local em todas as vizinhanças usadas e, no melhor caso, um mínimo

global.

Este trabalho apresenta a heurística ILS-VRPCD proposta para resolver o VRPCD. Esta é baseada em ILS e utiliza um VND na fase de busca local. A principal característica deste algoritmo é fato de explorar apenas o espaço viável de soluções.

A fase de geração de solução inicial do ILS-VRPCD é dada pelo método *GeraPoolInicial*. Este é responsável por gerar o *pool* inicial de soluções. O pseudocódigo apresentado no Algoritmo 4 define este processo. Na linha 1, o conjunto de soluções é inicializado. No laço das linhas 2–6, o conjunto de soluções é definido, onde cada solução s é gerada pela heurística construtiva 2S-NI, definida na Seção 3.1, na linha 3. Em seguida, na linha 4, a solução s' é gerada por um algoritmo VND. Este algoritmo explora as vizinhanças geradas pelas buscas locais *exchange* e *realocate*. Na linha 5, a solução s' é inserida no *pool*. Ao final do processo, na linha 7, o conjunto de soluções *Pop* é retornado.

Algoritmo 4: Função GeraPoolInicial().

```
1 Pop ← ∅;
2 repita
3 |   Construa uma solução s com a heurística 2S-NI;
4 |   s' ← VND(s);
5 |   Insira s' em Pop;
6 até k < n;
7 retorna Pop;
```

A fase *busca local* do ILS-VRPCD é realizada por um algoritmo VND. Este foi implementado utilizando as buscas locais inter-rotas *insertion*, *swap(1,1)*, *swap(2,1)* e *drop route*, definidas na Seção 3.2.2. Um processo de intensificação é realizado, pela busca local *ReInsertion Intraroute*.

Na fase de *perturbação* do ILS-VRPCD são executados os dois mecanismos definidos na Seção 3.2.4. A perturbação *Split* é executada após λ iterações do algoritmo sem que a melhor solução corrente seja atualizada, enquanto *Random-Exchange* é executada nas demais iterações. Estes mecanismos bem sincronizados possibilitam diversificar de forma eficiente sob o espaço de soluções.

O *critério de aceitação* é verificado após cada nova solução ótima local ser definida. Uma solução é aceita como novo ponto de busca do algoritmo se seu custo for no máximo $\alpha\%$ maior que o custo da solução corrente da iteração anterior. O ajuste ineficiente do parâmetro $\alpha\%$ compromete o processo de diversificação do método proposto.

O Algoritmo 5 ilustra a heurística ILS-VRPCD. Na linha 1 o conjunto *Pop* de soluções é gerado. Na linha 2, uma solução s é selecionada aleatoriamente de *Pop*. Na linha 3, a busca local *Insertion* é executada sob a solução s. As variáveis auxiliares são inicializadas na linha 4. No laço das linhas 5–20 executa-se sucessivamente *perturbação* e VND sob a solução s' para gerar um novo ótimo local s''. Em seguida, na linha 11, a solução s'' é refinada pelo operador *ReInsertion Intraroute*. O critério de aceitação é avaliado na linha 12. O conjunto *Pop* de soluções é atualizado sempre que uma nova solução de melhora for definida (linha 13), para isso a nova solução s' é inserida em *Pop* e a solução de pior custo deste é removida. Nas linhas 14-17, a solução $s*$ é atualizada ou a variável auxiliar para a escolha da perturbação é incrementada. Na linha 18, após λ iterações, uma nova solução corrente s' é selecionada aleatoriamente de *Pop*. Por fim, a solução $s*$ é retornada na linha 21. Os parâmetros definidos para o algoritmo são: o critério de parada, dado em função do tempo máximo de execução do laço principal do algoritmo, o valor limite do critério de aceitação (α), o número de iterações (λ) que a perturbação *Random-Exchange* é executada consecutivamente e que uma nova solução corrente é selecionada de *Pop* e a taxa de perturbação (φ).

Algoritmo 5: Heurística ILS-VRPCD

```
1  Pop ← GeraPoolInicial();
2  Selecione uma solução aleatória s de Pop;
3  s', s* ← BuscaLocal(Insertion, s);
4  i, it ← 0;
5  enquanto critério de parada não for satisfeito faça
6      se i = 0 então
7          s ← Perturbação(Split,s', φ);
8      senão
9          s ← Perturbação(Random-Exchange,s', φ);
10     s'' ← VND_CD(s);
11     s ← BuscaLocal(ReInsertionIntraRoute, s'');
12     s' ← CritérioDeAceitação(s', s, α);
13     Atualize a solução s' em Pop;
14     se f(s') ⩽ f(s*) então
15         s* ← s';
16     senão
17         i ← i + 1;
18     se i = λ então
19         i ← 0;
20         Selecione a solução aleatória s' de Pop;
21 retorna s*;
```

Uma solução corrente s' é escolhida de Pop quando a execução da heurística ILS-VRPCD estagna. Isto funciona como um mecanismo de "*restart*". Entretanto, ao invés de retornar para uma solução aleatória no espaço de busca, a heurística seleciona uma solução em Pop, que por sua vez possui melhor custo que a média das soluções geradas aleatoriamente. Dado que os métodos de busca local implementados são aleatórios, voltar para uma posição sabidamente boa e seguir uma direção diferente possibilita uma diversificação eficiente durante processo de busca.

Capítulo 4

Matheurística para o VRPCD

Os problemas de roteamento podem ser modelados como um problema de Particionamento de Conjuntos (SP, do inglês *Set Partitioning*) [Subramanian, 2012]. Seja M um conjunto finito e não vazio e L um conjunto de subconjuntos de M, em que a cada elemento $j \in L$ é associado um custo $t_j \in \mathbb{N}$. SP consiste em encontrar uma coleção de elementos de L, que é uma partição de M, onde o somatório dos custos associados a estes elementos seja mínimo. Ao se modelar o VRP em termos das restrições do SP, cada elemento em M representa um consumidor (ou fornecedor) e cada elemento do conjunto L representa uma rota. O objetivo desta abordagem é compor uma solução viável dada pela escolha ótima de um subconjunto de rotas em L que atendam todos os consumidores (ou fornecedores). Esta ideia vem sendo explorada em diversos trabalhos na literatura do VRP, sendo alguns dos mais recentes os de Alvarenga et al. [2007], Viana [2007], Brandão & Vasconcelos [2010] e Subramanian [2012].

4.1 SPILS-VRPCD

A matheurística SPILS-VRPCD, proposta para resolver o VRPCD, integra a heurística ILS-VRPCD, descrita no Capítulo 3, e um modelo de Programação Linear Inteira baseado em SP. Este algoritmo é dividido em duas partes. Na primeira, (i) um conjunto de rotas L é gerado pelas soluções definidas na fase de busca local da heurística ILS-VRPCD. Na segunda, (ii) uma nova solução é gerada pela resolução de um modelo de SP, dado o conjunto de rotas L. A motivação deste método reside no fato de soluções boas possuírem semelhanças estruturais em termos, por exemplo, de número de arestas ou sequência de nós visitados em uma rota. Assim, manter um conjunto de rotas dos mínimos locais, e periodicamente definir uma nova solução a partir deste, gera um processo eficiente de intensificação no processo de busca.

Seja k o número de veículos, R o conjunto de requisições e $L = L_p \cup L_d$ o conjunto de rotas, onde $L_p = \{l_1, l_2 ..., l_m\}$ é o conjunto de rotas de coleta e $L_d = \{l_1, l_2 ..., l_m\}$ o conjunto de rotas de entrega. Seja ainda o custo $t_j \in \mathbb{N}$ associado a cada rota $l_j \in L$ ($t_j = \sum_{(i,j) \in l_j} c_{ij}$) e as seguintes constantes: $\lambda_{i,j}$, que é igual a 1 se o fornecedor p_i da requisição $r_i \in R$ foi visitado pela rota de coleta $l_j \in L_p$, ou 0 caso contrário; $\delta_{i,j}$, que é igual a 1 se o consumidor d_i da requisição $r_i \in R$ foi visitado pela rota de entrega $l_j \in L_d$, ou 0 caso contrário. As variáveis de decisão são: x_j que assume valor 1 se a rota $l_j \in L_p$ é selecionada entre as rotas de coleta, ou 0 caso contrário; y_j que é igual a 1 se a rota $l_j \in L_d$ é escolhida entre as rotas de entrega, ou 0 caso contrário. A formulação baseada em SP para o VRPCD é descrita a seguir.

$$Minimizar \sum_{j \in L_p} t_j x_j + \sum_{j \in L_d} t_j y_j \tag{4.1}$$

Sujeito a:

$$\sum_{j \in L_p} \lambda_{ij} x_j = 1, \qquad \forall i \in R \tag{4.2}$$

$$\sum_{j \in L_d} \delta_{ij} y_j = 1, \qquad \forall i \in R \tag{4.3}$$

$$\sum_{j \in L_p} x_j \leq k, \tag{4.4}$$

$$\sum_{j \in L_d} y_j \leq k, \tag{4.5}$$

$$x_j, y_j \in \{0, 1\}, \qquad \forall j \in L \tag{4.6}$$

A função objetivo (4.1) minimiza o somatório dos custos ao se escolher a melhor combinação de rotas dos conjuntos L_p e L_d. As restrições (4.2) e (4.3) são as restrições originais da formulação de SP, que garantem que cada requisição $i \in R$ seja transportada por uma única rota de coleta e uma única rota de entrega. As restrições (4.4) e (4.5) impõem que a solução dada pelo modelo de SP tenha no máximo k rotas de coleta e k rotas de entrega. Por fim, as restrições (4.6) definem o domínio das variáveis de decisão.

O Algoritmo 6 ilustra o pseudocódigo do SPILS-VRPCD. Na linha 1 o conjunto de soluções iniciais *Pop* é criado, conforme descrito na Seção 3.3. O modelo de programação matemática *modelSP* é construído na linha 2 com as rotas que formam as soluções contidas em *Pop* (rotas contidas no conjunto L). Na linha 3, uma solução s

Algoritmo 6: Matheurística SPILS-VRPCD

```
1  Pop ← GeraPoolInicial();
2  modelSP ← DefinaModeloSP(Pop);
3  Selecione a solução aleatória s de Pop;
4  s', s* ← BuscaLocal(Insertion, s);
5  i, it ← 0;
6  s^sp ← s ;
7  enquanto critério de parada for não satisfeito faça
8      se i = 0 então
9          s ← Perturbação(Split,s', φ);
10     senão
11         s ← Perturbação(Random–Exchange,s', φ);
12     s'' ← VND_CD(s);
13     s ← BuscaLocal(ReInsertionIntraRoute, s'');
14     s' ← CritérioDeAceitação(s', s, α);
15     Atualize a solução s' em Pop;
16     se f(s') ⩽ f(s*) então
17         s* ← s';
18     senão
19         i ← i + 1;
20     se (altPop) então
21         Adicione as colunas de s' em modelSP;
22         s^sp ← Resolva(modelSP);
23         Viabilize a solução s^sp;
24         se f(s^sp) ⩽ f(s*) então
25             s* ← s^sp;
26             s' ← s^sp;
27     se i = λ então
28         i ← 0;
29         Selecione uma solução aleatória s' de Pop;
30 retorna s*;
```

é selecionada aleatoriamente de *Pop*. Na linha 4, a busca local *insertion* é executada sob s. As variáveis auxiliares são inicializadas nas linhas 5 e 6. No laço das linhas 7–29, executa-se sucessivamente perturbação e VND sob a solução s para gerar um novo ótimo local s'. Na linha 13, a busca local *ReInsertion Intraroute* é executada. O critério de aceitação é avaliado na linha 14. Uma nova solução s' é aceita se seu custo não for maior que $\alpha\%$ do custo da solução s. *Pop* é atualizado com a solução s' na linha 15. Uma solução s^p com o maior custo de *Pop* é substituída por s' se esta possuir menor custo, ou seja se $f(s') \leqslant f(s^p)$). Nas linhas 16-19, a solução $s*$ é atualizada ou

a variável auxiliar para a escolha da perturbação é incrementada. Caso o conjunto *Pop* seja atualizado por uma solução melhorante durante o processo de busca os passos das linhas 20–29 são executados. Na linha 21, as rotas da nova solução s' são incorporadas ao modelo *modelSP*. Na linha 22, o modelo *modelSP* é resolvido e a solução s^{sp} é definida. A viabilidade da solução s^{sp} é avalizada na linha 23. As soluções s^* e s' são atualizadas nas linhas 24-26. Nas linhas 27-29, após λ iterações, uma nova solução corrente s' é selecionada aleatoriamente do conjunto *Pop*. Por fim, a melhor solução encontrada é retornada na linha 30. No modelo de programação matemática definido o número de veículos (k) é dado pelo número de rotas da melhor solução encontrada ($s*$) até a iteração corrente. Os parâmetros definidos para o algoritmo são: o critério de parada, dado em função do tempo máximo de execução do laço principal do algoritmo, o valor limite do critério de aceitação (α), o número de iterações (λ) que a perturbação *Random-Exchange* é executada consecutivamente e que uma nova solução corrente é selecionada de *Pop* e a taxa de perturbação (φ).

SPILS-VRPCD possui dois conjuntos de rotas, o *Pop* e o conjunto de rotas de *modelSP*. O primeiro, mantém um conjunto com número constante de soluções viáveis para o problema, que é atualizado a cada iteração do método. Neste, cada nova solução melhorante substitui a solução com maior custo de função objetivo no conjunto. Já no *modelSP*, o número de rotas não é constante, porém é limitado. Quando o número de rotas de *modelSP* extrapola um dado limite, este é reinicializado com as rotas do conjunto *pop*. No entanto, um algoritmo de eliminação de rotas também poderia ser utilizado.

Uma característica da matheurística SPILS-VRPCD é que o modelo *modelSP* trata apenas do roteamento. Neste, as rotas de coleta e entrega são tratadas separadamente. Assim, ao constituir uma solução de VRPCD, pode-se gerar soluções inviáveis devidos às restrições de consolidação. Para tanto, a cada nova solução obtida pela resolução do modelo *modelSP* sua viabilidade deve ser avaliada (linhas 22 e 23 do Algoritmo 6). No processo de viabilização de uma solução, requisições são movidas de um veículo, cujas rotas de coleta/entrega que não atendem as restrições consolidação, para outros veículos. Este processo realiza movimentos semelhantes aos realizados pela busca local *Insertion*, em que os nós fornecedores e consumidores de uma requisição r_j são movidos de um veículo de origem v_i para outro veículo de destino v_j, tanto nas rotas de coleta quanto nas rotas entrega.

Capítulo 5

Experimentos Computacionais

Experimentos computacionais foram realizados em um computador com processador Intel® Core™ 2 Duo com 3.0 GHz e 4 GB de memória RAM, executando sob o sistema operacional Linux (kernel 3.0.0-12). Apenas um CPU core foi utilizado. Os algoritmos foram implementados em C++ e compilados com GCC 4.4.3. A implementação da matheurística SPILS-VRPCD faz uso do ILOG CPLEX na versão 12.1.

As instâncias de teste utilizadas foram propostas em Wen et al. [2009]. Elas são adaptadas de instâncias reais que pertencem à empresa dinamarquesa de consultoria em logística Transvision, instalada em Copenhagen. São dados cinco grupos de instâncias com 30, 50, 100, 150 e 200 requisições, nas quais a capacidade dos veículos é de 33 unidades. O tempo fixo de preparo do veículo no CD (A) é igual a 10 minutos e o tempo necessário para a consolidação de cada unidade de produto (B) igual a 1 minuto.

Este Capítulo é organizado da seguinte forma. Os resultados das heurísticas construtivas são apresentados na Seção 5.1. Os experimentos da ILS-VRPCD são descritos na Seção 5.2. Nesta além de resultados quanto à melhor solução e médias, é avaliado o impacto que o tempo de espera no CD tem no custo da solução. Na seção 5.3 são apresentados os experimentos da matheurística SPILS-VRPCD.

5.1 Experimentos com Heurísticas Construtivas

Neste experimento, são avaliadas as três heurísticas construtivas propostas. Os resultados deste experimento são apresentados na Tabela 5.1. As duas primeiras colunas apresentam os nomes e o número de nós (fornecedores e consumidores) das instâncias. A coluna 3 apresenta o custo da solução obtida pela heurística 2S-NI. Os resultados da heurística 2S-NN são apresentados na coluna 4. Na última coluna são descritos

os resultados da heurística 2S-S. Em negrito são apresentados os melhores resultados obtidos entre as heurísticas construtivas.

Tabela 5.1: Comparação das heurísticas construtivas para as instâncias de [Wen et al., 2009].

Instância	$\|P \cup D\|$	2S-NI *Custo*	2S-NN *Custo*	2S-S *Custo*
30a	60	**5.098,84**	5.205,85	5.543,23
30b	60	**5.982,83**	6.379,63	6.353,45
30c	60	**6.110,08**	6.412,32	7.070,27
30d	60	**5.185,08**	5.306,07	5.674,43
30e	60	6.654,59	**6.489,50**	6.725,90
50a	100	8.228,90	8.908,94	**7.755,55**
50b	100	**9.088,57**	9.852,60	9.788,70
50c	100	**9.342,72**	9.403,14	9.451,25
50d	100	8.958,73	9.614,89	**9.373,85**
50e	100	10.550,68	**9.471,94**	10.387,76
100a	200	**15.474,25**	16.192,87	16.673,66
100b	200	**17.460,23**	18.721,94	19.018,27
100c	200	**17.150,43**	17.591,54	17.564,17
100d	200	**17.332,55**	17.352,67	18.045,62
100e	200	17.888,43	**17.220,96**	17.789,36
150a	300	23.869,40	**23.480,38**	24.823,49
150b	300	26.011,97	**25.960,52**	26.438,76
150c	300	25.498,07	**25.117,59**	25.298,47
150d	300	26.020,27	**25.768,36**	26.771,61
150e	300	25.677,83	25.157,76	**25.130,81**
200a	400	33.547,16	**33.383,42**	33.486,41
200b	400	**33.415,71**	33.676,18	34.987,06
200c	400	**32.335,96**	32.594,05	34.199,30
200d	400	**33.435,46**	33.653,01	35.619,95
200e	400	**32.641,42**	33.169,35	34.136,09
	Média	**18.118,40**	18.243,41	18.724,29

Pode-se observar que, embora a implementação das heurísticas construtivas apresentem a mesma complexidade assintótica, 2S-NI obteve os melhores resultados. Das 25 instâncias avaliadas a 2S-NI encontrou 14 melhores soluções, 2S-NN obteve melhores resultados em 8, enquanto a 2S-S encontrou melhores soluções para outras 3 instâncias. Para a instância 200e o custo da solução encontrada por 2S-NI foi 32.641,42, pela 2S-NN foi 33.169,35, enquanto que pela heurística 2S-S foi 34.136,09. Na média, o custo das soluções encontradas por 2S-NI foi de 18.118,40, enquanto 2S-NN e 2S-S encontram valores de média iguais a 18.243,41 e 18.724,29, respectivamente. Por encontrar na média os melhores resultados, a heurística 2S-NI foi escolhida para gerar o conjunto de soluções iniciais para os algoritmos ILS-VRPCD e SPILS-VRPCD.

5.2 Experimentos com Heurística ILS-VRPCD

O algoritmo ILS-VRPCD utiliza os seguintes parâmetros: critério de aceitação (α), taxa de perturbação (φ), número de iterações (λ) para escolha aleatoriamente de uma solução do conjunto Pop e tamanho (η) do conjunto *Pop*. Baseado em testes preliminares os seguintes valores para estes parâmetros foram selecionados: $\alpha = 6\%$ do custo da solução corrente, $\varphi = 4\%$ do número de requisições, $\lambda = 10$ iterações e $\eta = 20$ soluções.

No primeiro experimento para ILS-VRPCD, o critério de parada foi definido em 3.000 segundos, como sugerido por Tarantilis [2012]. Foram realizadas 25 execuções independentes do algoritmo para cada instância, variando a semente do gerador de números aleatórios. A Tabela 5.2 mostra a comparação de ILS-VRPCD com as melhores heurísticas da literatura para as instâncias de Wen et al. [2009]. As colunas 1 e 2 descrevem as instâncias utilizadas. A coluna 3 informa o limite inferior (LB) para cada instância, dado pelo valor da relaxação linear do modelo 2-VRPTW, descrito em Wen et al. [2009]. O custo da melhor solução definida pela heurística de Wen et al. [2009] ($f(s)$) e o *gap* percentual ($(f(s) - LB)/LB$) deste em relação ao valor de LB são apresentado nas colunas 4 e 5, respectivamente. Os mesmos resultados são apresentados para o algoritmo de Tarantilis [2012] nas duas colunas seguintes. O custo da melhor solução obtida pela heurística ILS-VRPCD e o *gap* desta em relação ao valor de LB são apresentado nas colunas 8 e 9, respectivamente. O valor médio para 25 execuções da heurística proposta é mostrado na coluna 10. Por fim, a última coluna fornece o *gap* entre o valor médio do algoritmo ILS-VRPCD e o valor de LB. A média dos resultados é apresentada separadamente para cada grupo de instâncias. Como pode ser observado Tarantilis [2012] não informa em seus trabalhos os resultados para as instâncias 30a, 30b, 30c, 30d, 30e e 100a.

Pode-se observar que ILS-VRPCD mostrou ser mais eficiente que as heurísticas existentes para VRPCD. Em particular, 12 dos melhores resultados foram obtidos, com *gaps* que variam de 0,59% a 3,92% ao se comparar os melhores resultados obtidos em relação ao valor de LB. Na média, o custo da melhor solução obtida por ILS-VRPCD foi de 2.57%, enquanto o de Wen et al. [2009] foi 2.73%. Em relação à média, o algoritmo obteve *gaps* que variam de 0,93% a 6,91%.

No segundo experimento, é avaliado o impacto que o tempo de espera no CD tem no custo da solução. Neste experimento, para um conjunto com 4 instâncias (150a, 150e, 200d e 200e), foram realizadas 25 execuções do algoritmo ILS-VRPCD variando o tempo fixo de preparo do veículo (A) e o tempo necessário para carregar ou descarregar cada unidade demandada por produtos de uma requisição no CD (B). O gráfico da

Tabela 5.2: Comparação de ILS-VRPCD com as melhores heurísticas da literatura para as instâncias de [Wen et al., 2009]

			[Wen et al., 2009]		[Tarantilis, 2012]		ILS-VRPCD			
Instância	$\|P \cup D\|$	LB	Melhor custo	Gap *(%)*	Melhor custo	Gap *(%)*	Melhor custo	Gap *(%)*	Média	Gap (Média-LB)%
30a	60	3.757,04	3.884,7	3,4	-	-	**3.884,71**	3,4	3.889,63	3,53
30b	60	4.795,65	4.824,1	0,59	-	-	**4.824,08**	0,59	4.840,07	0,93
30c	60	4.968,30	5.112,4	2,9	-	-	**5.107,79**	2,81	5.125,33	3,16
30d	60	3.708,37	3.850,0	3,82	-	-	**3.848,62**	3,78	3.898,63	5,13
30e	60	4.913,24	5.014,3	2,06	-	-	**5.013,87**	2,05	5.073,45	3,26
Média	-	4.428,52	4.537,10	2,55	-	-	4.535,81	2,53	4.565,42	3,20
50a	100	6.340,90	6.471,9	2,07	**6.450,28**	1,72	6.462,06	1,91	6.500,50	2,52
50b	100	7.201,89	**7.410,6**	2,90	7.428,54	3,15	7.436,06	3,25	7.464,34	3,64
50c	100	7.241,05	7.330,6	1,24	**7.311,77**	0,98	7.365,89	1,72	7.502,33	3,61
50d	100	6.887,93	7.050,3	2,36	**7.021,39**	1,94	7.050,22	2,36	7.079,80	2,79
50e	100	7.347,54	7.516,8	2,3	**7.451,42**	1,41	7.501,06	2,09	7.557,75	2,86
Média	-	7.003,90	7.156,0	2,17	7.132,68	1,84	7.163,06	1,84	7.220,94	3,08
100a	200	12.555,57	12.860,8	2,43	-	-	**12.851,08**	2,35	13.407,22	6,78
100b	200	14.200,48	14.526,1	2,29	**14.405,52**	1,44	14.418,09	1,53	14.981,53	5,50
100c	200	13.631,24	13.967,8	2,47	13.889,22	1,89	**13.823,33**	1,41	1.4427,80	5,84
100d	200	13.395,33	13.763,3	2,75	**13.564,23**	1,26	13.644,20	1,86	14.321,34	6,91
100e	200	13.745,60	14.212,7	3,40	**14.059,62**	2,28	14.064,78	2,32	14.589,18	6,14
Média	-	13.505,60	13.866,14	2,67	-	-	13.760,30	1,90	14.345,41	6,24
150a	300	19.012,02	**19.537,3**	2,76	19.638,04	3,29	19.546,00	2,81	19.944,63	4,91
150b	300	20.371,08	20.974,8	2,96	20.922,27	2,71	**20.906,28**	2,63	21.197,41	4,06
150c	300	19.419,55	20.126,5	3,64	20.019,50	3,09	**20.010,14**	3,04	20.372,38	4,91
150d	300	20.013,37	**20.549,4**	2,68	20.600,33	2,93	20.595,17	2,91	20.831,70	4,09
150e	300	19.141,66	19.848,5	3,69	19.782,00	3,35	**19.737,29**	3,11	20.021,36	4,60
Média	-	19.591,54	20.207,30	3,15	20.192,43	3,07	20.158,98	2,90	20.473,50	4,51
200a	400	26.538,53	27.324,4	2,96	27.397,31	3,24	**27.287,22**	2,82	27.774,08	4,66
200b	400	26.722,88	27.637,7	3,42	**27.582,87**	3,22	27.770,12	3,92	27.984,97	4,72
200c	400	25.607,31	**26.358,6**	2,93	26.425,29	3,19	26.450,67	3,29	26.787,42	4,61
200d	400	26.969,42	27.749,7	2,89	27.818,77	3,15	**27.707,66**	2,74	28.197,80	4,55
200e	400	25.776,01	**26.620,6**	3,28	26.704,81	3,60	26.700,21	3,59	27.182,11	5,46
Média	-	26.322,83	27.138,20	3,10	27.185,81	3,28	27.183,18	3,27	27.585,28	4,80
Média		-	-	2,73	-	-	-	**2,57**	-	4,37

Figura 5.1 ilustra a relação entre o tempo de consolidação e o custo das soluções para VRPCD. Como pode ser observado, a medida que se aumenta os tempos de espera no CD, os custos das soluções do problema também aumentam. Isto porque o processo de consolidação deve ser realizado dentro de um intervalo de tempo que permita que os produtos sejam entregues aos consumidores. Caso o tempo de consolidação no CD seja insuficiente o transporte direto dos fornecedores aos consumidores sem consolidação é realizado.

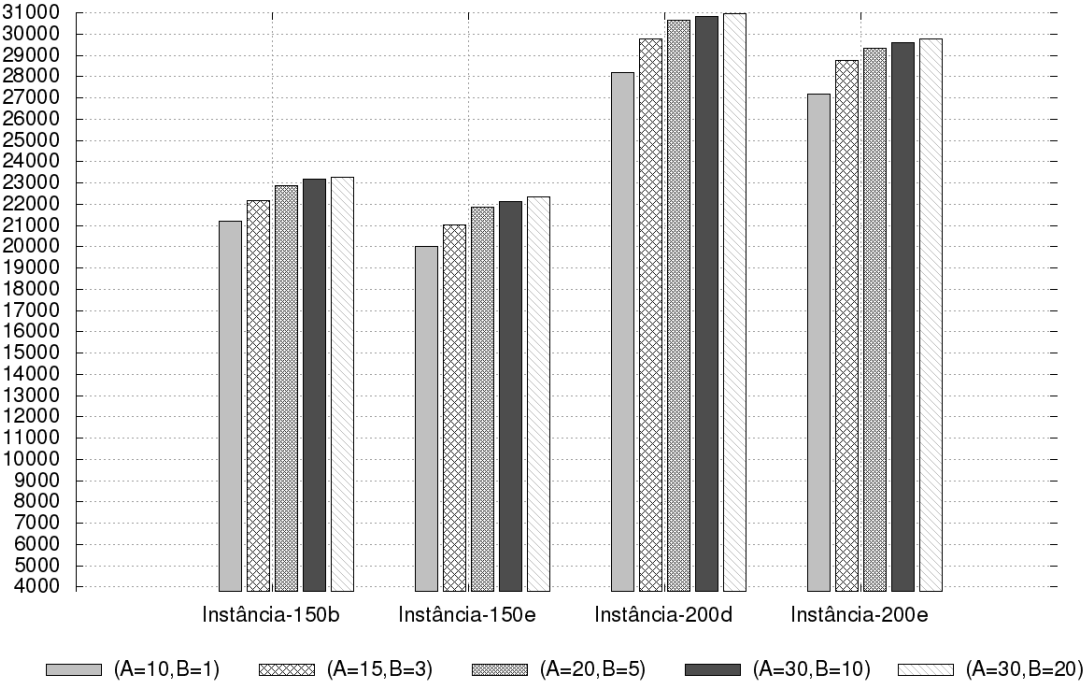

Figura 5.1: Gráfico de avaliação do impacto do tempo de espera no CD em relação ao custo das soluções para VRPCD, variando-se os tempos de preparo do veículo (A) e o tempo necessário para carregar ou recarregar cada unidade de produto (B).

5.3 Experimentos com Matheurística SPILS-VRPCD

A matheurística SPILS-VRPCD utiliza os seguintes parâmetros: critério de aceitação (α), taxa de perturbação (φ), número de iterações (λ) para escolha aleatoriamente de uma solução do conjunto Pop e tamanho (η) do conjunto Pop. Baseado em testes preliminares os seguintes valores para estes parâmetros foram selecionados: $\alpha = 5\%$ do custo da solução corrente, $\varphi = 7\%$ do número de requisições, $\lambda = 10$ iterações e $\eta = 10$ soluções.

No primeiro experimento para SPILS-VRPCD, assim como na heurística ILS-VRPCD, o critério de parada foi definido em 3.000 segundos. Foram realizadas 25 execuções independentes do algoritmo para cada instância, variando a semente do gerador de números aleatórios. Os resultados deste experimento são apresentados na Tabela 5.3. As colunas 1 e 2 descrevem as instâncias utilizadas. A coluna 3 informa o limite inferior LB para cada instância, conforme descrito em Wen et al. [2009]. Os melhores resultados, a média e o *gap* entre a melhor solução encontrada e LB para SPILS-VRPCD são apresentados nas colunas 4, 5 e 6, respectivamente. O desvio relativo $((f(s) - f(s^h))/f(s^h))$ entre os melhores resultados da heurística ILS-VRPCD

($f(s^h)$), encontrados na coluna 8 da Tabela 5.2, e SPILS-VRPCD ($f(s)$) são mostrados na coluna 7. O desvio relativo entre os melhores resultados da literatura, obtidos pelos algoritmos de Wen et al. [2009] e Tarantilis [2012] é apresentado nas colunas 4 e 6 da Tabela 5.2, e o custo das melhores soluções encontradas pela matheurística proposta é dado na coluna 8.

Tabela 5.3: Comparação da matheurística SPILS-VRPCD com as melhores heurísticas da literatura para as instâncias de [Wen et al., 2009]

			SPILS-VRPCD				
Instância	$\|P \cup D\|$	LB	*Melhor custo*	Média	*Gap* (%)	*Desvio em relação a ILS-VRPCD (%)*	*Desvio em relação a* [Wen et al., 2009] *e* [Tarantilis, 2012](%)
30a	60	3757,04	3.884,71	3.885,33	3,40	0,00	0,00
30b	60	4795,65	4.824,08	4.836,89	0,59	0,00	0,00
30c	60	4968,3	5.107,79	5.121,17	2,81	0,00	-0,09
30d	60	3708,37	3.843,87	3.888,34	3,65	-0,12	-0,16
30e	60	4913,24	5.010,31	5.023,66	1,98	-0,07	-0,08
50a	100	6340,9	6.455,33	6.464,91	1,80	-0,10	0,08
50b	100	7201,89	7.433,08	7.439,16	3,21	-0,04	0,06
50c	100	7241,05	7.326,70	7.488,22	1,18	-0,53	0,20
50d	100	6887,93	7.030,90	7.056,81	2,08	-0,27	0,14
50e	100	7347,54	7.463,70	7.512,17	1,58	-0,50	0,16
100a	200	12555,57	12.775,75	12.875,39	1,75	-0,59	-0,66
100b	200	14200,48	14.406,60	14.492,91	1,45	-0,08	0,01
100c	200	13631,24	13.819,90	13.993,76	1,38	-0,02	-0,50
100d	200	13395,33	13.629,65	13.806,87	1,75	-0,11	0,48
100e	200	13745,6	14.060,37	14.160,99	2,29	-0,03	0,01
150a	300	19012,02	19.513,46	19.809,34	2,64	-0,17	-0,12
150b	300	20371,08	20.862,92	21.079,80	2,41	-0,21	-0,28
150c	300	19419,55	19.997,84	20.255,39	2,98	-0,06	-0,11
150d	300	20013,37	20.552,06	20.802,14	2,69	-0,21	0,01
150e	300	19141,66	19.726,03	19.921,56	3,05	-0,06	-0,28
200a	400	26538,53	27.119,09	27.295,43	2,19	-0,62	-0,75
200b	400	26722,88	27.623,85	27.742,76	3,37	-0,53	0,15
200c	400	25607,31	26.400,41	26.514,32	3,10	-0,19	0,16
200d	400	26969,42	27.690,84	28.048,35	2,67	-0,06	-0,21
200e	400	25776,01	26.516,91	27.028,27	2,87	-0,69	-0,39
	Média	14.170,47	14.523,05	14.661,76	2,36	-0,21	-0,09

Pode-se observar que SPILS-VRPCD superou ILS-VRPCD em todos os casos, exceto nas instâncias 30a, 30b e 30c, onde ambos os métodos obtiveram os mesmos resultados. Na média, a diferença entre estes dois métodos foi de 0.21%. Em comparação com a literatura a heurística ILS-VRPCD obteve os melhores resultados em 12 das instâncias, enquanto SPILS-VRPCD encontrou 14 novas melhores soluções. Em relação aos melhores resultados apresentados nos trabalhos de Wen et al. [2009] e Tarantilis [2012] o algoritmo SPILS-VRPCD obteve melhorias que variam entre 0.09% e 0.75%. O *gap* médio entre a matheurística proposta e o *LB* foi 2.36%.

Para pequenas instâncias, com 30 requisições, pode-se verificar que tanto os algoritmos apresentados nesta dissertação quanto aqueles descritos na literatura obtiveram

resultados semelhantes. Já para as instâncias com 50 e 100 requisições o algoritmo Tarantilis [2012] obtive os melhores resultados. Enquanto os algoritmos ILS-VRPCD e SPILS-VRPCD obtiveram melhores resultados para as instâncias de grande porte, ou seja, instâncias com mais de 100 requisições.

No segundo experimento, os algoritmos ILS-VRPCD e SPILS-VRPCD foram induzidos a parar sempre que uma solução com um custo menor ou igual a um dado valor alvo foi encontrada. Este teste foi realizado para quatro instâncias com 200 requisições (200b, 200c, 200d e 200e). Para estas instâncias, foram realizadas 100 execuções dos algoritmos variando o valor da semente do gerador de números aleatórios. O valor alvo foi definido pela média de 10 execuções de cada instância em um tempo de 5 minutos para o algoritmo ILS-VRPCD. Então, o gráfico TTT plot (do inglês, *Time-To-Target value plots*) [Aiex et al., 2006] foi criado. O TTT plot é uma técnica para comparar o desempenho de diferentes algoritmos estocásticos para um dado problema de otimização. Dada uma instância e um valor alvo, um TTT plot exibe a probabilidade de um algoritmo encontrar uma solução, cujo custo é tão bom quanto o alvo em um dado tempo em execução. Os TTT plots para as instâncias avaliadas são apresentados nas Figuras 5.2 a 5.5. Nestes, pode-se notar uma convergência mais rápida do SPILS-VRPCD em direção ao valor alvo.

Na Figura 5.2, o TTT plot para a instância 200b é ilustrado. Pode-se observar que com probabilidade de 90% a matheurística SPILS-VRPCD encontrar uma solução tão boa quanto o alvo em 200 segundos, enquanto a ILS-VRPCD gasta mais de 600 segundos. A probabilidade de SPILS-VRPCD encontrar uma solução tão boa quanto o alvo é sempre maior do que a ILS-VRPCD.

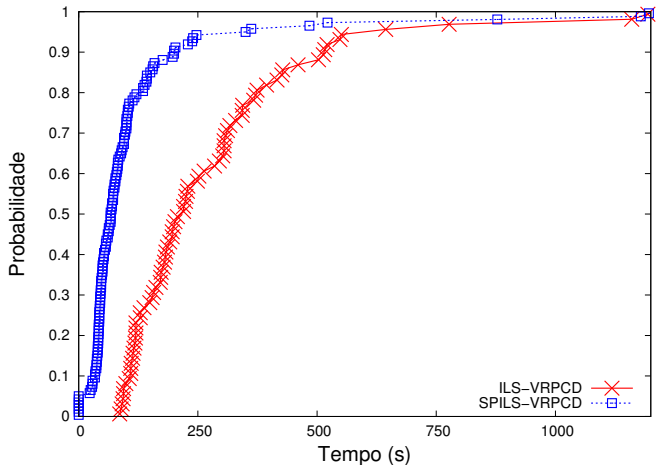

Figura 5.2: TTT plot para a instância 200b. Neste experimento, a probabilidade da matheurística SPILS-VRPCD encontrar uma solução tão boa quanto o alvo é sempre maior que o da heurística ILS-VRPCD. SPILS-VRPCD encontra a solução alvo em menos de 350 segundos com probabilidade igual 96%, enquanto a ILS-VRPCD alcança valores semelhantes ao alvo em 620 segundos.

O TTT plot para a instância 200c é apresentado na Figura 5.3. Onde pode ser observado que a probabilidade de SPILS-VRPCD encontrar uma solução tão boa quanto o alvo é sempre maior do que ILS-VRPCD até os 750 segundos de execução. Além do mais, com probabilidade próxima a 100%, SPILS-VRPCD não gastou mais de 1100 segundos para encontrar uma solução tão boa quanto o alvo, enquanto ILS-VRPCD passou de 1472 segundos.

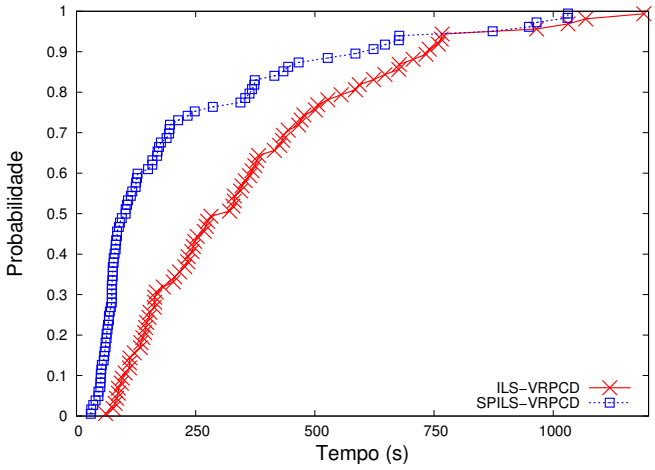

Figura 5.3: TTT plot para a instância 200c. No experimento realizado, a heurística ILS-VRPCD define uma solução com custo igual a alvo em 1200, porém com probabilidade de 92% este encontra o alvo em 760 segundos. Já a matheurística SPILS-VRPCD define a solução alvo com probabilidade próxima a 100% em 1100 segundo. O grau de convergência da matheurística é maior do que o de ILS-VRPCD.

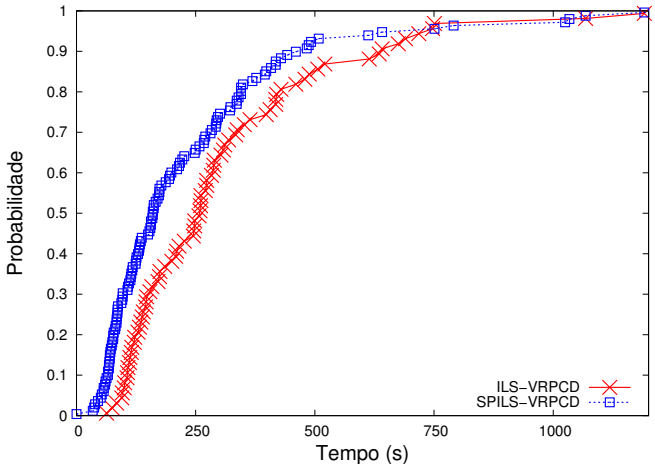

Figura 5.4: TTT plot para a instância 200d. Neste experimento a evolução da heurística ILS-VRPCD se mostrou bastante similar ao da matheurística SPILS-VRPCD. A probabilidade de se definir uma solução tão boa quanto o alvo a partir dos 750 segundos para ambos os algoritmos é relativamente próxima. Porém, o algoritmo SPILS-VRPCD apresenta uma convergência mais acentuada até os 500 segundos.

A Figura 5.4 apresenta o TTT plot para a instância 200d. Neste experimento o comportamento da heurística ILS-VRPCD se mostrou bastante similar ao da matheu-

rística SPILS-VRPCD. A probabilidade de se definir uma solução tão boa quanto o alvo a partir dos 750 segundos para ambos os algoritmos é relativamente próxima.

Finalmente, o TTT plot para a instância 200e é apresentado na Figura 5.5. Pode-se observar que com probabilidade de 90% a matheurística SPILS-VRPCD encontrar uma solução tão boa quanto o alvo em 230 segundos, enquanto a ILS-VRPCD gasta mais de 650 segundos. Com probabilidade próxima a 100%, SPILS-VRPCD define uma solução tão boa quanto alvo em 1050 segundos, enquanto ILS-VRPCD encontra uma solução próxima ao alvo em 1350 segundos.

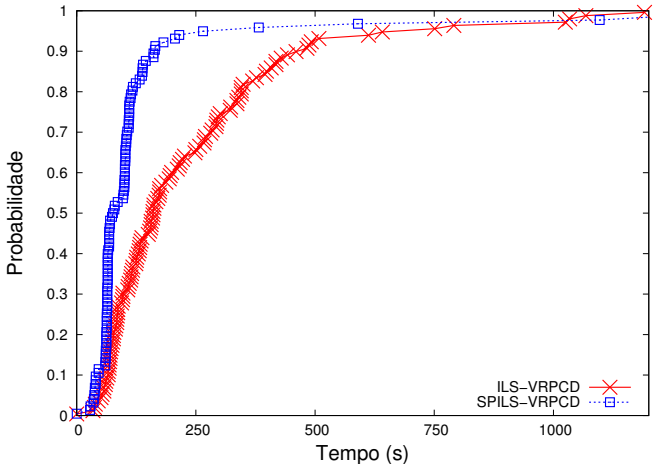

Figura 5.5: TTT plot para a instância 200e. Neste experimento, a probabilidade da SPILS-VRPCD encontrar uma solução tão boa quanto o alvo é maior do que a de ILS-VRPCD. Com probabilidade igual a 90%, SPILS-VRPCD define uma solução tão boa quanto alvo em 230 segundos, enquanto o ILS-VRPCD encontra uma solução com mesma probabilidade em 550 segundos.

Capítulo 6

Conclusão e Trabalhos Futuros

Nesta dissertação foi abordado o Problema de Roteamento de Veículos com *Cross-Docking* (VRPCD, do inglês *Vehicle Routing Problem with Cross-Docking*). Este problema consiste em minimizar os custos de transporte ao se coletar produtos em diversos fornecedores e entregá-los aos consumidores via um único centro de consolidação (CD), onde o processo de consolidação é realizado. Em VRPCD, os processos de coleta e entrega são dependentes, uma vez que é permito realizar a transferência de produtos entre os veículos no CD.

Para tratar o problema foram apresentadas três heurísticas construtivas, uma heurística baseada na metaheurística ILS, chamada ILS-VRPCD, e a matheurística SPILS-VRPCD, que integra ILS-VRPCD e um modelo de Programação Linear Inteira baseado no problema de Particionamento de Conjuntos. As heurísticas construtivas propostas são baseadas em heurísticas clássicas da literatura de VRP e são capazes de gerar soluções de boa qualidade com baixo custo computacional. A heurística ILS-VRPCD e a matheurística SPILS-VRPCD exploram o conjunto viável de soluções e encontram soluções comparáveis às da literatura.

Os experimentos realizados comprovaram a eficiência dos algoritmos implementados. Além disso, verificou-se que o tempo para realizar o processo de consolidação gera um grande impacto na atividade de roteamento. Com isso, a medida que os tempos de troca no CD aumentam, os custos das soluções do problema também aumentam. Isso porque o processo de consolidação se torna caro, o que pode inviabilizar o transporte de produtos via CD, devido as restrições de janela de tempo. Assim, os ganhos ao se incorporar o processo de consolidação a um problema de roteamento ocorrem principalmente nos casos onde o tempo necessário para realizar consolidação é baixo em relação às janelas de tempo dos consumidores e do CD.

Em trabalhos futuros, uma possibilidade seria buscar garantir a otimalidade das

soluções para a versão de VRPCD estudada. Nesta linha métodos exatos, como Branch-and-Bound, Geração de Colunas, Branch-and-Cut e Branch-and-Price, poderiam ser utilizados. Na linha de heurísticas, ainda podem ser investigados algoritmos baseados em Simulated Annealing, Late Acceptance Hill Climbing, GRASP com Path Relinking, Colônia de Formigas, etc.

Estudar outras versões de VRPCD é outra possibilidade de trabalhos futuros. Nessa linha, problemas que abordam (i) coleta e entrega fracionária de produtos, (ii) transporte direto ou indireto de produtos através do CD, (iii) frotas de veículos heterogêneos, ou ainda, (iv) contextos onde exista mais de um centro de consolidação, contribuiriam para a literatura do problema. Todas estas versões podem ser resolvidas com algoritmos adaptados daqueles apresentados nesta dissertação.

Referências Bibliográficas

Aiex, R. M.; Resende, M. G. C. & Ribeiro, C. C. (2006). Tttplots: A perl program to create time-to-target plots. *Optimization Letters*, 1:10–1007.

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

Apte, U. M. & Viswanathan, S. (2003). Strategic and technological innovations in supply chain management. *International Journal of Manufacturing Technology and Management*, 4(3-4):264–282.

Aragao, M. P. & Uchoa, E. (2003). Integer program reformulation for robust branch-and-cut-and-price algorithms. Em *Proceedings of the Conference Mathematical Programming in Rio: A Conference in Honour of Nelson Maculan, Rio de Janeiro*, pp. 56–61.

Assis, L. P. (2007). Algoritmos para o problema de roteamento de veículos com coleta e entrega simultâneas. Dissertação de mestrado, Universidade Federal de Minas Gerais.

Balakrishnan, N. (1993). Simple heuristics for the vehicle routing problem with soft time windows. *Journal of the Operational Research Society*, 44:279–287.

Bartholdi, J. J. & Gue, K. R. (2004). The best shape for a crossdock. *Transportation Science*, 38(2):235–244.

Berbeglia, G.; Cordeau, J.-F.; Gribkovskaia, I. & Laporte, G. (2007). Static pickup and delivery problems: A classification scheme and survey. *TOP: Sociedad Espanola de Estadistica e Investigacion Operativa, Springer Verlag*, 15(1):1–31.

Bookbinder, M. G. J. H. (2004). Cross-docking and its implications in location-distribution systems. *Journal of Business Logistics*, 25(2):199–228.

Brandão, H. O. & Vasconcelos, G. (2010). A hybrid search method for the vehicle routing problem with time windows. *Annals of Operations Research*, 180:125–144.

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

Bräysy, O. & Gendreau, M. (2005b). Vehicle routing problem with time windows, part II: Metaheuristics. *Transportation Science*, 39:119–139.

Chen, P.; Guo, Y.; Lim, A. & Rodrigues, B. (2006). Multiple crossdocks with inventory and time windows. *Computers and Operations Research*, 33:43–63.

Clarke, G. & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581.

Cordeau, J.-F.; Gendreau, M.; Hertz, A.; Laporte, G. & Sormany, J.-S. (2005). New heuristics for the vehicle routing problem. Em Langevin, A. & Riopel, D., editores, *Logistics Systems: Design and Optimization*, pp. 279–297. Springer.

Cordeau, J.-F.; Gendreau, M. & Laporte, G. (1997). A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30(2):105–119.

Cordeau, J.-F.; Laporte, G. & Mercier, A. (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society*, 52:928–936.

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

Dethloff, J. (2001). Vehicle routing and reverse logistics: The vehicle routing problem with simultaneous delivery and pick-up. *OR Spectrum*, 23(1):79–96.

Dror, M.; Laporte, G. & Trudeau, P. (1994). Vehicle routing with split deliveries. *Discrete Applied Mathematics*, 50(3):239–254.

Fukasawa, R.; Longo, H.; Lysgaard, J.; de Aragão, M. P.; Reis, M. L.; Uchoa, E. & Werneck, R. F. F. (2006). Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, 106(3):491–511.

Gendreau, M.; Laporte, G. & Potvin, J.-Y. (2001). *Metaheuristics for the Capacitated VRP*, capítulo Metaheuristics for the capacitated VRP, pp. 129–154. Society for Industrial and Applied Mathematics.

Gendreau, M. & Potvin, J.-Y., editores (2010). *Handbook of Metaheuristics*, volume 146. International Series in Operations Research & Management Science, 2 edição.

Golden, B.; Raghavan, S. & Wasil, E. A. (2008). *The vehicle routing problem : Latest advances and new challenges*. Operations Research/Computer Science Interfaces Series, 43. Springer.

Ibaraki, T.; Imahori, S.; Nonobe, K.; Sobue, K.; Uno, T. & Yagiura, M. (2008). An iterated local search algorithm for the vehicle routing problem with convex time penalty functions. *Discrete Applied Mathematics*, 156(11):2050–2069.

Jayaraman, V. & Ross, A. (2003). A simulated annealing methodology to distribution network design and management. *European Journal of Operational Research*, 144(3):629–645.

Laporte, G. (1992). The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3):345–358.

Lee, Y. H.; Jung, J. W. & Lee, K. M. (2006). Vehicle routing scheduling for cross-docking in the supply chain. *Computers and Industrial Engineering*, 51:247–256.

Liao, C.-J.; Lin, Y. & Shih, S. C. (2010). Vehicle routing with cross-docking in the supply chain. *Expert Systems with Applications*, 37(10):6868–6873.

Lourenço, H. R.; Martin, O. C. & Stutzle, T. (2010). Iterated local search: Framework and applications. Em Glover, F.; Kochenberger, G. & Hillier, F., editores, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research and Management Science*, capítulo 12, pp. 363–398. Springer, 2ª edição.

Min, H. (1989). The multiple vehicle routing problem with simultaneous delivery and pickup points. *Transportation Research Part A: General*, 23(5):377–386.

Montané, F. A. T.; Filho, V. J. M. F. & Galvão, R. D. (1997). Determinação de rotas para empresas de entrega expressa. *Pesquisa Operacional*, 17:107–135.

Mosheiov, G. (1998). Vehicle routing with pick-up and delivery: Tour-partitioning heuristics. *Computers and Industrial Engineering*, 34(3):669--684.

Musa, R.; Arnaout, J.-P. & Jung, H. (2010). Ant colony optimization algorithm to solve for the transportation problem of cross-docking network. *Computers and Industrial Engineering*, 59(1):85–92.

Nagarajan, V. & Ravi, R. (2012). Approximation algorithms for distance constrained vehicle routing problems. *Networks*, 59(2):209–214.

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

Napolitano, M. (2002). *Making the move to cross docking: A practical guide to planning, designing, and implementing a cross dock operation*. Warehousing Education and Research Council, J.E. Gross and Associates.

Ombuki, B.; Ross, B. J. & Hanshar, F. (2006). Multi-objective genetic algorithms for vehicle routing problem with time windows. *Applied Intelligence*, 24:2006.

Parragh, S. N.; Doerner, K. F. & Hartl, R. F. (2008). A survey on pickup and delivery problems. *Journal fur Betriebswirtschaft*, 58:21–51.

Penna, P. H. V.; Subramanian, A. & Ochi, L. S. (2011). An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *Journal of Heuristics*, pp. 1–32.

Pisinger, D. & Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers and Operations Research*, 34(8):2403 – 2435.

Ratliff, H. D.; Vate, J. V. & Zhang, M. (1999). Network design for load-driven drossdocking systems. Relatório técnico, The Logistics Institute, Georgia Institute of Technology, Atlanta.

Ribas, S.; Subramanian, A.; Coelho, I. M.; Ochi, L. S. & Souza, M. J. F. (2010). A hybrid search method for the vehicle routing problem with time windows. *Annals of Operations Research*, 180:125–144.

Rosenkrantz, D. J.; E.Stearns, R. & II, P. M. L. (1977). An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing*, 6(3):563–581.

Salani, M. (2005). Branch-and-price algorithms for vehicle routing problems. Dissertação de mestrado, Facoltà di Scienze Matematiche, Fisiche e Naturali, Dipartimento di Tecnologie dell' Informazione, Università Degli Studi Di Milano.

Santos, F. A.; Cunha, A. S. & Mateus, G. R. (2010). Modelos de otimização para o problema de roteamento de veículos com cross-docking. *XLII Simpósio Brasileiro de Pesquisa Operacional*, 1:1–12.

Santos, F. A.; Mateus, G. R. & da Cunha, A. S. (2011a). A branch-and-price algorithm for a vehicle routing problem with cross-docking. *Electronic Notes in Discrete Mathematics*, 37(0):249–254.

Santos, F. A.; Mateus, G. R. & da Cunha, A. S. (2011b). A novel column generation algorithm for the vehicle routing problem with cross-docking. Em *Network Optimization*, volume 6701 of *Lecture Notes in Computer Science*, pp. 412–425. Springer.

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

Subramanian, A. (2012). *Heuristic, Exact and Hybrid Approaches for Vehicle Routing Problems*. Tese de doutorado, Universidade Federal Fluminense.

Tarantilis, C. (2012). Adaptive multi-restart tabu search algorithm for the vehicle routing problem with cross-docking. *Optimization Letters*, pp. 1–14.

Viana, F. H. F. (2007). Algoritmo para o problema de roteamento dinâmico de veículos com janelas de tempo e tempos de viagem variáveis. Dissertação de mestrado, Universidade Federal de Minas Gerais.

Vidal, T.; Crainic, T. G.; Gendreau, M. & Prins, C. (2013). A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers and Operations Research*, 40(1):475–489.

Wen, M.; Larsen, J.; Clausen, J.; Cordeau, J.-F. & Laporte, G. (2009). Vehicle routing with cross-docking. *Journal of the Operational Research Society*, 60:6868–6873.

Witt, C. E. (1998). Crossdocking: Concepts demand choice. *Material Handling Engineering*, 53(7):44–50.

Yu, W. & Egbelu, P. J. (2008). Scheduling of inbound and outbound trucks in cross docking systems with temporary storage. *European Journal of Operational Research*, 184:377–296.