UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES
PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

PEDRO YURI ARAUJO LIMA ALVES

Problema de estoque e roteirização com demanda estocástica e janelas de tempo: uma abordagem utilizando relaxação lagrangeana

São Paulo
2018

PEDRO YURI ARAUJO LIMA ALVES

Problema de estoque e roteirização com demanda estocástica e janelas de tempo: uma abordagem utilizando relaxação lagrangeana

Dissertação apresentada à Escola de Artes, Ciências e Humanidades da Universidade de São Paulo para obtenção do título de Mestre em Ciências pelo Programa de Pós-graduação em Sistemas de Informação.

Área de concentração: Metodologia e Técnicas da Computação

Versão corrigida contendo as alterações solicitadas pela comissão julgadora em 23 de março de 2018. A versão original encontra-se em acervo reservado na Biblioteca da EACH-USP e na Biblioteca Digital de Teses e Dissertações da USP (BDTD), de acordo com a Resolução CoPGr 6018, de 13 de outubro de 2011.

Orientador: Profa. Dra. Karina Valdivia Delgado

São Paulo

2018

CATALOGAÇÃO-NA-PUBLICAÇÃO
(Universidade de São Paulo. Escola de Artes, Ciências e Humanidades. Biblioteca)
CRB 8 - 4936

Alves, Pedro Yuri Araujo Lima
Problema de estoque e roteirização com demanda estocástica e janelas de tempo : uma abordagem utilizando relaxação lagrangeana / Pedro Yuri Araujo Lima Alves ; orientadora, Karina Valdivia Delgado. – 2018.
109 p. : il

Dissertação (Mestrado em Ciências) - Programa de Pós-Graduação em Sistemas de Informação, Escola de Artes, Ciências e Humanidades, Universidade de São Paulo.
Versão corrigida

1. Inteligência artificial. 2. Programação linear. 3. Estoques. 4. Logística. I. Delgado, Karina Valdivia, orient. II. Título

CDD 22.ed.– 006.3

Dissertação de autoria de Pedro Yuri Araujo Lima Alves, sob o título **"Problema de estoque e roteirização com demanda estocástica e janelas de tempo: uma abordagem utilizando relaxação lagrangeana"**, apresentada à Escola de Artes, Ciências e Humanidades da Universidade de São Paulo, para obtenção do título de Mestre em Ciências pelo Programa de Pós-graduação em Sistemas de Informação, na área de concentração Metodologia e Técnicas da Computação, aprovada em 23 de março de 2018 pela comissão julgadora constituída pelos doutores:

Profa. Dra. Karina Valdivia Delgado

Instituição: Universidade de São Paulo

Presidente

Prof. Dr. Eduardo Candido Xavier

Instituição: Universidade estadual de Campinas

Prof. Dr. Fernando Fagundes

Instituição: Universidade de São Paulo

Prof. Dr. Daniel de Angelis Cordeiro

Instituição: Universidade de São Paulo

Agradecimentos

Este espaço é dedicado àqueles que contribuíram de alguma forma para que esse projeto fosse realizado. Mesmo não sendo possível nomear a todos, há alguns que não posso deixar de expressar o meu apreço e agradecimento sincero.

À Profa. Dra. Karina Valdivia Delgado, minha orientadora, por acreditar no meu potencial e por estar disposta a me acompanhar em uma trabalho que foge de sua área de atuação. Agradeço pela paciência, pela orientação impecável e pelo incentivo constante durante todo esse período de execução do projeto.

Ao Prof. Dr. Alexandre da Silva Freire, pela infinita disponibilidade, por estar sempre disposto a me ouvir, tirar minhas dúvidas e por me conduzir em áreas nas quais meu conhecimento era simplório.

Ao Prof. Dr. Fernando Fagundes Ferreira, pelo apoio, paciência e disponibilidade dos recursos necessários para a finalização desse trabalho.

Ao Prof. Dr. Valdinei Freire da Silva pela colaboração inestimável.

Ao Prof. Dr. Carlos Eduardo Ferreira, pelas sugestões e contribuições fornecidas para que esse trabalho se tornasse ainda melhor.

A meus pais, Rosangela e Carlos, que sempre acreditaram na minha capacidade e que me ensinaram a sempre buscar o melhor possível para mim e todos a minha volta.

Aos meus grandes amigos, Renan, Bruno, Talita, Thiago, Brunno e Tayse, pela paciência durante a "fase mestrado".

Aos meus amigos de profissão, Auro e Walter, pelo apoio, suporte e direcionamento que forneceram durante esse período de estudos.

A meus amigos do mestrado, Cleiton Alves e Ricardo Feitosa, pelos momentos trilhados juntos.

E por fim agradeço a Carolline, meu porto seguro, por ser tão importante na minha vida. Sempre me apoiando e me colocando para cima nos momentos mais difíceis. Obrigado pelo seu companheirismo, compreensão, paciência, amizade, alegria e amor. Cada momento e cada palavra tornaram possível a concretização desse trabalho.

"Your time is limited, so don't waste it living someone else's life. Don't be trapped by dogma which is living with the results of other people's thinking. Don't let the noise of others' opinions drown out your own inner voice. And most important, have the courage to follow your heart and intuition. They somehow already know what you truly want to become. Everything else is secondary. (...)

Stay hungry, stay foolish."

(Steve Jobs, 2005)

Resumo

ALVES, Pedro Yuri Araujo Lima. **Problema de estoque e roteirização com demanda estocástica e janelas de tempo:** uma abordagem utilizando relaxação lagrangeana. 2018. 109 f. Dissertação (Mestrado em Ciências) – Escola de Artes, Ciências e Humanidades, Universidade de São Paulo, São Paulo, 2018.

Fornecedores necessitam atender a demanda de seus clientes da forma mais adequada possível e mantendo a qualidade de seu serviço, porém em muitos casos essa demanda é desconhecida. Esse problema pode ser modelado como um problema de roteirização e estoque com demanda estocástica o qual inclui o controle de estoque, transporte do produto e decisões de agendamento da entrega. Existem vários trabalhos na literatura para resolver esse problema, porém nenhum deles lida com janela de tempo de atendimento, capacidade máxima de estoque tanto no cliente quanto no depósito e o nível de confiança de atendimento individualizado para cada cliente. O objetivo principal deste trabalho é propor um novo algoritmo baseado em otimização matemática para lidar com esse problema mais realista. Além disso, este trabalho tem como objetivo secundário melhorar o algoritmo de estado da arte baseado em otimização matemática, visando encontrar soluções com um menor tempo computacional e custo. Foram realizados experimentos com instâncias sintéticas com 15 até 50 clientes, as quais são geradas aleatoriamente, e com uma instância real, baseada na experiência profissional no mercado empresarial e em cenários reais de distribuição na cidade de São Paulo.

Palavras-chaves: Problema de roteamento e estoque; Janela de tempo; Demanda estocástica; Relaxação lagrangeana.

Abstract

ALVES, Pedro Yuri Araujo Lima. **Inventory and routing problem with stochastic demand and time windows:** an approach using lagrangean relaxation. 2018. 109 p. Dissertation (Master of Science) – School of Arts, Sciences and Humanities, University of São Paulo, São Paulo, 2018.

Providers need to supply the demand of their clients as optimally as possible and maintaining the quality of their service, however in many cases this demand is unknown. This problem can be modeled as a inventory routing problem with stochastic demand, which includes inventory control, product transportation and delivery scheduling decisions. There are several papers in the literature to solve this problem, but none of them deals with service time window, maximum stock capacity for both the customer and the depot and individualized confidence level for each costumer. The main objective of this work is to propose a new algorithm based on mathematical optimization to deal with this more realistic problem. In addition, this work has as secondary objective to improve the state of the art algorithm based on mathematical optimization, aiming to find solutions with a lower computational time and cost. Experiments were performed with synthetic instances with 15 to 50 clients, which are randomly generated, and with a real instance, based on professional experience in the business market and in real distribution scenarios in the city of São Paulo.

Keywords: Inventory routing problem; Time window; Stochastic demand; Lagrangian Relaxation.

Lista de figuras

Lista de algoritmos

Lista de tabelas

Lista de abreviaturas e siglas

VRP	Problema de Roteamento de Veículos (Vehicle Routing Problem).
IRP	Problema de Roteamento e Inventários (Inventory Routing Problem).
SIRP	Problema de Roteamento e Inventários com demanda estocástica (Stochastic Inventory Routing Problem).
MDP	Processo de decisão Markoviano (Markov Decision Process).
LI	Limite inferior.
LS	Limite superior.
CWS	Algoritmo de economias de Clarke & Wright (Clarke & Wright Savings).
RLI	Problema de Inventário.
RLR	Problema de Roteamento.
VRPTW	Problema de roteamento de veículo com janela de tempo (Vehicle Routing Problem with Time Window).
GRASP	Busca adaptativa randomizada gulosa (Greedy Randomized Adaptive Search Procedure).
TSP	Problema do caixeiro viajante (Traveling Salesman Problem).
MCS	Simulação de Monte-Carlo (Monte Carlo Simulation).
IRPTW	Problema de Roteamento e Inventário com Janela de Tempo (Inventory Routing Problem with Time Window).
PFIH	Heurística de inserção para frente (Push Forward Insertion Heuristic).
SIRPTW	Problema de Roteamento e Inventários com demanda estocástica e janelas de tempo (Stochastic Inventory Routing Problem with Time Window).
PIME	Programação Inteira Mista e Estocástica.

Ct	Método de Rahim adaptado com Centroide.
SA	Método de Rahim adaptado com Simulated Annealing.
MC	Método de Rahim adaptado com Monte Carlo.
C&W	Clarke & Wright.
CWTW	Método para SIRPTW com Clarke & Wright.
SATW	Método para SIRPTW com Simulated Annealing.
JR	Janela restritiva.
JM	Janela mista.
JS	Janela sem restrição.

Lista de símbolos

Z_P	Valor da função objetivo de P.
x	Vetor de tamanho $n \times 1$.
b	Vetor de tamanho $m \times 1$.
e	Vetor de tamanho $k \times 1$.
$\mathring{A}$	Matriz $m \times n$.
D	Matriz $k \times n$.
P	Problema genérico de otimização.
LR_μ	Problema Lagrangeano.
D^μ	Problema Dual.
$Z_L(\mu)$	Valor da função objetivo de LR_μ.
$\mu = (\mu_1, ..., \mu_k)$	Vetor de multiplicadores lagrangianos.
X	Conjunto de soluções para o problema LR_μ.
x^*	Solução ótima para o problema P.
Z_D	Valor da função objetivo de D^μ.
g^k	matriz de subgradiente.
x^{μ_k}	Solução ótima para o problema LR_μ.
s^k	Tamanho do passo para atualização de μ.
ω	Parâmetro de atualização do passo.
$S = \{0, 1, \ldots, n\}$	Conjunto de vértices do grafo.
$A = \{(i, j) \mid i, j \in S; i \neq j\}$	Conjunto de arestas do grafo.
c_{ij}	Custo da viagem entre os clientes.
d_i	Demanda de entrega de cada cliente.

R_v Rota de um veículo.

m Conjunto de veículos disponíveis para a operação.

s_{ij} Economia de custo.

ζ Conjunto de clientes.

Θ_{ij} Custo de rota.

T Tamanho do horizonte de planejamento.

$H = \{1, ..., T\}$ O horizonte de planejamento.

h Período dentro do horizonte de planejamento.

H^+ Horizonte de planejamento mais o período zero.

C_j Capacidade de estoque do cliente.

R_{rh} Quantidade de produto que será entregue ao fornecedor.

I_{jh} Níveis de estoque do fornecedor e dos clientes.

d_{jh} Demanda dos clientes.

$V = \{1, ..., K\}$ Conjunto de veículos.

k_v Capacidade de cada veículo.

q_{jh} Quantidade de produto a ser entregue.

b_j Inicio da janela de tempo do cliente.

e_j Fim da janela de tempo do cliente.

t_j Horário de chegada e início de atendimento do cliente.

tq_{jh} Tempo de entrega do cliente.

t_j^d Horário de saída do cliente.

$P_j(t_j)$ Penalidade de não atendimento de janela de tempo.

x_{ijv} Valor binário quando o veículo visita o cliente j logo após de i.

z_α	Valor de distribuição normal padrão.
$r = 0$	Indice do depósito no grafo.
v	Veículo pertencente ao conjunto V.
S^2	Combinação de todos os pares $(i, j) \in S$.
$D_j = E(d_{jh})$	Média da demanda do cliente.
σ_j	Desvio padrão da demanda do cliente.
η_{jh}	Custo de armazenamento.
ψ_v	Custo operacional do veículo.
δ_v	Custo de viagem do veículo.
ν_v	Velocidade média do veículo.
θ_{ij}	Duração da viagem.
Λ_{jh}	Demanda estocástica do cliente no período.
Γ_{ijhv}	Total de custo de entrega e transporte.
x_{ijhv}	Valor binário quando o veículo visita o cliente j após i no período h.
y_{hv}	Valor binário de utilização do veículo.
Q_{ijhv}	Quantidade do produto restante no veículo.
Z_{RLI}	Valor da função objetivo de RLI.
Z_{RLR}	Valor da função objetivo de RLR.
Ck_{LI}	Custo da solução de RLI + RLR.
$\mathcal{P}$	Conjunto de parâmetros.
τ_h	Total de horas de um período.
T	Parâmetro de temperatura.
SR_h	Conjunto de clientes que tem entrega no período h.

α	Taxa de decrescimento da temperatura T.
M_0	Unidade de tempo.
β	Taxa de crescimento.
$Time$	Contador de tempo.
$MaxTime$	Limite de tempo.
T_b	Valor de temperatura da melhor solução.
T_k	Valor da temperatura atual da iteração.
τ	Taxa de diminuição da temperatura T_k.
T_r	Valor da temperatura de recomeço.
$S^{'}$	Solução vizinha a S.
$N_2(S)$	Conjunto de soluções vizinhas a S.
Δ	Valor da diferença do custo entre as soluções $S^{'}$ e S.
S_b	Melhor solução encontrada.
$nReset$	Contador de recomeços.

Sumário

[1] De acordo com a Associação Brasileira de Normas Técnicas. NBR 6023.

1 Introdução

Tendo em vista o aumento da competitividade global, pode-se notar que a logística tem um papel de suma importância na gestão de negócios empresariais. A prioridade da área de logística desses fornecedores é satisfazer os clientes, buscando sempre meios de melhorar e, principalmente, reduzir seus custos nos processos logísticos (BELFIORE; COSTA; FÁVERO, 2006). Os fornecedores preocupam-se com aspectos da cadeia de suprimentos que inclui o armazenamento, a determinação das rotas e as quantidades a serem entregues aos seus clientes, enfrentando diversos problemas para diminuir os custos nesse processo.

Problemas de otimização de cada um desses aspectos da cadeia de suprimentos foram apresentados ao longo dos anos e sua evolução natural é a junção desses problemas para resolvê-los em conjunto. Nos modelos mais tradicionais, os próprios clientes controlam seus estoques, isto é, quando o cliente acredita que é necessário o reabastecimento, o mesmo realiza o pedido ao fornecedor considerando suas necessidades naquele momento. Esse problema é modelado como um Problema de Roteamento de Veículos (em inglês Vehicle Routing Problem – VRP).

A integração do gerenciamento de estoque, VRP e decisões do agendamento das entregas é conhecida como Problema de Roteamento e Inventários (em inglês Inventory Routing Problem – IRP) (COELHO; CORDEAU; LAPORTE, 2013). No IRP são os fornecedores que decidem quando deverá ser reabastecido o estoque baseados na demanda de seus clientes, tendo como objetivo que os mesmos não fiquem sem o produto. Normalmente, IRPs trabalham com mais de um período de operação, considerando os impactos de suas decisões no futuro da distribuição (TOTH; VIGO, 2002).

O IRP tem por objetivo a otimização dos custos de estoque e custos que envolvem a quantidade, frequência de distribuição e as rotas adotadas para a realização da reposição de estoque. Dessa forma é necessária a coordenação entre o controle de estoque, transporte do produto e decisões de agendamento da entrega. O IRP é de grande auxílio para diminuir o custo de logística de uma empresa, pois promove eficiência na operação e melhoria na qualidade do serviço.

Com relação ao tempo de quando a informação da demanda se torna conhecida, os IRPs podem ser classificados em determinísticos ou estocásticos. Se essa informação

está totalmente disponível durante a tomada de decisão, isto é, no início do horizonte de planejamento, o problema IRP é determinístico. Caso somente a distribuição de probabilidade da demanda é conhecida, o problema IRP é estocástico (em inglês Stochastic Inventory Routing Problem – SIRP). Os trabalhos propostos para solução do SIRP tem o mesmo objetivo que em sua versão determinística, porém com a dificuldade de atender a estocasticidade e os futuros valores desconhecidos da demanda.

As soluções para SIRP foram categorizadas em (COELHO; CORDEAU; LAPORTE, 2013) como soluções heurísticas, baseados em programação dinâmica e baseados em otimização robusta. A maioria dos trabalhos estão classificados como algoritmos heurísticos, como por exemplo em (FEDERGRUEN; ZIPKIN, 1984a), (HVATTUM; LØKKETANGEN; LAPORTE, 2009), (HUANG; LIN, 2010) e (JUAN et al., 2014). Outros trabalhos estão baseados em programação dinâmica e modelam o problema como um processo de decisão Markoviano (em inglês Markov Decision Process – MDP), como em (KLEYWEGT; NORI; SAVELSBERGH, 2002) e (ADELMAN, 2004). Porém, essa última abordagem é impraticável para instâncias de SIRP de tamanho médio e grande (HVATTUM; LØKKETANGEN; LAPORTE, 2009).

A última classe são dos trabalhos baseados em otimização robusta. Essa abordagem é apropriada para trabalhar com incertezas, quando não há informação disponível sobre os parâmetros da distribuição de probabilidade da demanda (COELHO; CORDEAU; LAPORTE, 2013).

Um dos trabalhos mais recentes na literatura que usa programação matemática é (RAHIM et al., 2014). Esse trabalho utiliza relaxação de Lagrange para: (i)relaxar as restrições do problema original que são consideradas complexas para o modelo; (ii)estimar limites inferiores e superiores para a solução; e (iii) atualizar esses valores para obter uma solução viável e de menor custo. Essa abordagem demonstra um bom desempenho computacional, obtendo uma solução viável em poucos minutos. Para melhorar ainda mais o tempo computacional e a qualidade das soluções apresentadas, este trabalho adaptou diferentes heurísticas para encontrar soluções com custo menor. Além disso, o modelo proposto por Rahim et al. (2014) não é totalmente adaptável para situações de problemas em empresas reais, pois não considera janelas de tempo de atendimento do cliente, a capacidade máxima de estocagem dos clientes e do depósito, bem como o nível de confiança do atendimento não é individualizado para cada cliente. Para resolver essas limitações, este trabalho também pretende modificar o problema de otimização incluindo novas restrições.

1.1 Objetivos

Este trabalho tem como objetivo melhorar o algoritmo baseado em otimização robusta proposto em (RAHIM et al., 2014) para resolver o Problema de Inventários e Roteamento com demandas estocásticas, visando encontrar soluções com custo menor e diminuir o tempo computacional da solução. Além disso, pretende-se estender essa solução para problemas mais realistas que consideram as seguintes variáveis: capacidade máxima de estoque, tanto no cliente quanto no depósito, confiança de atendimento individual para cada cliente e janela de tempo de atendimento do cliente.

Para atender o objetivo principal, consideramos como objetivos secundários a construção de um gerador de instâncias padronizadas conforme a modelagem de geração proposta em (YU; CHEN; CHU, 2008).

1.2 Justificativa

O SIRP é um problema facilmente encontrado no mundo real, como na indústria de componentes químicos, na indústria de óleo e gás e no ambiente marítimo, por isso uma boa solução do problema é necessária. Além de uma boa solução, é importante também que o tempo computacional para obter essa solução seja baixo, por esse motivo cria-se a necessidade de estender os algoritmos já existentes na literatura.

Um dos primeiros trabalhos sobre SIRP é o proposto por Federgruen e Zipkin (1984a), sendo que a partir dele, diversos autores abordaram esse tema na literatura. Porém, ainda hoje, muitas empresas fazem esse trabalho à mão ou utilizam regras muito básicas para atendimento dos clientes. Por exemplo, a política de sempre abastecer todo o estoque do cliente, ou pedidos emergenciais quando o estoque está no fim, sendo necessário um reabastecimento não programado de emergência (BENOIST et al., 2009). Além disso, apesar dos problemas de estoque e roteirização serem de natureza estocástica, a maioria dos trabalhos encontrados na literatura resolvem o problema de maneira determinística (BELFIORE; COSTA; FÁVERO, 2006). Essa opção é preferida, pois modelos estocásticos, em geral, requerem uma grande quantidade de dados históricos e um tratamento estatístico adequado, que não é uma tarefa simples.

Coelho, Cordeau e Laporte (2013) classificaram as soluções para o SIRP em três grupos: os algoritmos heurísticos, algoritmos que usam programação dinâmica e algoritmos que utilizam otimização robusta. A maioria dos trabalhos existentes na literatura são heurísticos, porém, o problema com esse tipo de abordagem é que os algoritmos são baseadas na distribuição da probabilidade da demanda o que pode influenciar na qualidade da solução caso a distribuição de probabilidade da demanda se distancie da distribuição inicial. Já as abordagens que usam programação dinâmica para o resolver o problema modelado como MDPs só conseguem resolver problemas pequenos, pois devem encontrar as ações ótimas para cada estado (COELHO; CORDEAU; LAPORTE, 2013).

Nos últimos anos, métodos que combinam otimização robusta ou o algoritmo branch-and-cut com outras técnicas têm se mostrado cada vez mais presentes na literatura (COELHO; CORDEAU; LAPORTE, 2013).

Um dos trabalhos mais recentes que abrange diversas características e variáveis que estão presentes em problemas da vida real é o proposto por Rahim et al. (2014). Esse trabalho utiliza otimização robusta para resolver o problema SIRP, desenvolvendo um algoritmo que usa a relaxação de Lagrange para encontrar limites inferiores e superiores para as soluções factíveis. Apesar dessa técnica encontrar soluções factíveis, seu modelo não é totalmente adaptável para situações de problemas com empresas reais. Por exemplo, no modelo proposto, não é considerada a variável de capacidade de estoque dos clientes, considerando que o estoque é sempre grande o suficiente para o reabastecimento do produto. Também não é considerado o nível de confiança de atendimento individual para cada cliente; nem janelas de tempo de atendimento do cliente.

1.3 Método

Visando atender o objetivo central do projeto, inicialmente, foi realizada uma revisão bibliográfica acerca do SIRP. Observando o crescimento dos trabalhos que abordam a otimização robusta como solução do SIRP e considerando que o algoritmo proposto por Rahim et al. (2014) é um dos mais completos em relação às variáveis incluídas, esse algoritmo foi selecionado e considerado como o estado-da-arte. Assim, foi realizada a implementação, teste e validação desse algoritmo.

Uma vez que não existe um benchmark de testes conhecido para o SIRP, foi modelado e desenvolvido um gerador de instâncias de testes o qual é utilizado nos experimentos. O algoritmo proposto por Rahim et al. (2014) foi implementado na linguagem Java com auxílio da ferramenta de otimização matemática Gurobi Optimizer[1].

Além disso, foram estudados meios para propor variações ao método de Rahim et al. (2014), o objetivo foi adaptar o método utilizado nesse trabalho. Os resultados das soluções propostas foram comparadas aos resultados obtidos por Rahim et al. (2014). Nessa comparação, foram utilizadas instâncias criadas do benchmark de testes de tamanho pequeno (15 clientes), médio (25 clientes) e grande (50 clientes). Para cada teste realizado foram analisados o custo final da solução, limites inferiores e superiores e tempo computacional.

Em seguida, foram estudadas possíveis extensões do SIRP, incluindo no problema as características de janela de tempo, capacidade máxima de estoque tanto no cliente quanto no depósito e o nível de confiança de atendimento individualizado para cada cliente. Os experimentos para essas extensões foram realizados considerando diferentes tamanhos de instancias (pequeno, médio e grande) e diferentes tipos de janelas de tempo.

1.4 Organização

O restante deste texto está organizado da seguinte forma. O Capítulo 2 salienta os principais conceitos do modelo de programação linear inteira e introduz os métodos de relaxação mais utilizados na literatura, que permitem obter limites da solução ótima dos problemas de programação inteira. O Capítulo 3 introduz o problema de roteamento de veículos, juntamente com alguns dos principais métodos para soluciona-lo. O Capítulo 4 apresenta os conceitos fundamentais para o entendimento do projeto incluindo a definição do IRP determinístico e estocástico e apresenta os algoritmos existentes.

Considerando os trabalhos estudados na revisão da literatura, foi possível observar a importância do problema SIRP na otimização logística em diversos ramos de negócios e o crescimento dos artigos relacionados. Diversos métodos foram propostos para encontrar uma melhor solução para esse problema de grande complexidade, sendo que um dos mais recentes é a técnica proposta em (RAHIM et al., 2014) que trabalha com uma modelagem de otimização robusta e aplica a relaxação de Lagrange. O Capítulo 5 descreve de forma detalhada esse algoritmo que é usado como base deste trabalho.

[1] Disponível em http://www.gurobi.com

O Capítulo 6 detalha as variações propostas para o algoritmo de Rahim para SIRP. No Capítulo 7, a principal contribuição do trabalho é apresentada, o modelo proposto é descrito detalhando as restrições adicionais, sua extensão para atender o problema de estoque e roteamento estocástico com janela de tempo e os algoritmos implementados para encontrar a solução desse modelo. O Capítulo 8 apresenta a descrição dos experimentos realizados juntamente com os resultados obtidos. Por fim, o Capítulo 9 apresenta a conclusão deste trabalho.

2 Métodos de relaxação para resolver problemas de programação inteira

A programação linear inteira, conhecida também como programação inteira, é um modelo de programação linear no qual algumas ou todas as variáveis do problema pertencem ao conjunto dos números inteiros. Um modelo de programação inteiro se trata de um problema NP-difícil, por esse motivo existem técnicas de relaxação do modelo para encontrar uma solução ótima ou quase ótima (WOLSEY, 1998).

A relaxação de Lagrange, a relaxação Surrogate, a relaxação Lagrange/Surrogate (*lagsur*), a relaxação combinada Lagrangeana-Surrogate e a relaxação de programação linear são as técnicas mais usadas para obter limites da solução ótima de problemas de programação inteira (SENNE; LORENA, 1999; NARCISO; LORENA, 1999; NARCISO; LORENA, ; LORENA; PEREIRA; SALOMÃO, 2003; ESPEJO; GALVÃO, 2002). Neste trabalho estamos interessados na relaxação de Lagrange, explicada a seguir.

Seja o problema genérico de otimização (P):

$$\begin{aligned} Z_P = min: \quad & cx \\ s.a: \quad & \mathring{A}x = b, \\ & Dx \leq e, \\ & x \geq 0 \ e \ inteiro, \end{aligned} \tag{1}$$

em que Z_P é o valor da função objetivo do problema; x, b e e são vetores de tamanho $n \times 1$, $m \times 1$ e $k \times 1$, respectivamente; $\mathring{A}$ é uma matriz $m \times n$ e D é uma matriz $k \times n$. As restrições de (P) são divididas em dois conjuntos, $\mathring{A}x = b$ e $Dx \leq e$. Considere que $\mathring{A}x = b$ seja a restrição complicadora do problema. O problema Lagrangeano (LR_μ) para o problema (P) irá considerar as restrições complicadoras de (P) e aplicar a relaxação Lagrangeana sobre elas. Essas restrições multiplicadas por um vetor μ, são adicionadas à função objetivo. Assim, (LR_μ) é formulado como:

$$\begin{aligned} Z_L(\mu) = min: \quad & cx + \mu(\mathring{A}x - b) \\ s.a: \quad & Dx \leq e, \\ & x \geq 0 \ e \ inteiro, \end{aligned} \tag{2}$$

em que $\mu = (\mu_1, ..., \mu_k)$ é um vetor de multiplicadores lagrangianos. O problema (LR_μ) é mais fácil de resolver do que seu problema original (P). É feita a suposição que (P) é um problema viável e o conjunto $X = \{x|Dx \leq e,\ x \geq 0\ e\ inteiro\}$ de soluções para o problema (LR_μ) é finito para todo μ, assim $Z_L(\mu)$ é finito.

É fácil provar que $Z_L(\mu) \leq Z_P$ (FISHER, 1981). Considere que a solução ótima para o problema (P) é x^*, então:

$$Z_L(\mu) \leq cx^* + \mu(\mathring{A}x^* - b) = cx^* = Z_P.$$

Nessa inequação, é considerada a definição de $Z_L(\mu)$, que $Z_P = cx^*$ e que $\mathring{A}x^* - b = 0$. Nesse caso, o multiplicador Lagrangeano μ pode ser positivo ou negativo.

Como demonstrado em (WOLSEY, 1998), se a restrição em (P) a ser relaxada for $\mathring{A}x \leq b$ no lugar de $Ax = b$, então é necessário que $\mu \geq 0$, assim temos:

$$Z_L(\mu) \leq cx^* + \mu(\mathring{A}x^* - b) \leq cx^* \leq Z_P,$$

pois $Z_P = cx^*$ e $\mathring{A}x^* - b \leq 0$. De maneira similar, para $\mathring{A}x \geq b$, a condição $Z_L(\mu) \leq Z_P$ se mantém caso seja garantido que $\mu \leq 0$.

Dessa forma, é possível utilizar o problema (LR_μ) para fornecer um limite inferior (LI) da solução ótima para o problema original (P). A qualidade do limite gerado depende do valor de μ. A melhor solução de μ será a solução ótima do problema dual:

$$Z_D = max: \quad Z_L(\mu) \tag{3}$$

Os métodos mais populares usados em aplicações de relaxação Lagrangeana para resolver o problema dual (D^μ) são: o método subgradiente, várias versões do método simplex implementados usando técnicas de geração de colunas e métodos de ajuste de multiplicadores (FISHER, 1981).

Estamos interessados no método subgradiente descrito a seguir. Esse método é uma generalização do método gradiente no qual o gradiente da função é substituído por um subgradiente para obter uma nova direção de busca. São realizadas buscas locais no gradiente da função e em cada ponto da busca calcula-se o subgradiente. A busca continua até não obter uma melhoria considerável.

Dado um valor inicial μ_0, uma sequência de multiplicadores de Lagrange é gerada pela equação:

$$\mu_{k+1} = \mu_k + s_k g^k,$$

em que g^k é o gradiente na iteração k, definido como:

$$g^k = \mathring{A}x^{\mu_k} - b,$$

x^{μ_k} é a solução ótima para (LR_μ) na iteração k e s_k é o tamanho do passo para atualização de μ. Esse tamanho é determinado considerando a seguinte regra:

$$s^k = \omega[Z^* - Z_D(\mu^k)]/\left\|g^k\right\|^2.$$

O parâmetro ω varia no intervalo $0 < \omega \leq 2$, começando com $\omega = 2$. Z^* que é um limite superior (LS) para a solução ótima de Z_D, pode ser encontrado aplicando uma heurística (i.e., encontrando uma solução viável para (P))(HELD; KARP, 1971). É importante destacar, que essa regra tem bom desempenho empírico, no entanto não garante a convergência do algoritmo para o valor ótimo da função (FISHER, 1981).

3 Problema de roteamento de veículos

Neste capítulo é abordado brevemente o problema de roteamento de veículos (VRP), apresentando os algoritmos mais utilizados na literatura e que serão utilizados nos algoritmos propostos neste trabalho.

3.1 Problema básico de VRP

O problema de roteamento de veículos é um problema de otimização combinatória e de programação inteira, em que uma solução viável é um conjunto de rotas para uma frota de veículos que atenda um conjunto de clientes. Um método para resolver esse problema deve buscar a solução ótima ou quase ótima do problema, ou seja, o conjunto ideal de rotas para uma frota de veículos para atender um determinado conjunto de clientes.

Esse problema foi inicialmente proposto em (DANTZIG; RAMSER, 1959) e uma instância do mesmo normalmente é definida como um grafo $G(S, A)$ em que $S = \{0, 1, \ldots, n\}$ é o conjunto de vértices, tal que o vértice 0 é a localização do depósito e $A = \{(i, j)|i, j \in S; i \neq j\}$ é o conjunto de arestas de ligação entre cada vértice. Nesse modelo c_{ij} é o custo da viagem entre os clientes i e j, a demanda de entrega de cada cliente é d_i, R_v é a rota de um veículo v e m é o conjunto de veículos disponíveis para a operação.

3.1.1 Algoritmo Clark & Wright Savings

O algoritmo de economias de Clarke & Wright (em inglês Clarke & Wright Savings – CWS) (CLARKE; WRIGHT, 1964) é um dos algoritmos heurísticos mais conhecidos para VRP e é aplicado principalmente a problemas nos quais o número de veículos não é fixo. O algoritmo monta uma lista de economias de custo obtidas pela união de duas rotas. A economia é obtida pela função $s_{ij} = c_{i0} + c_{0j} - c_{ij}$ no qual c_{ij} é o custo de viagem partindo de um cliente i a um cliente j. Inicialmente, o pior cenário possível é considerado, isto é o cenário em que os clientes i e j são visitados em rotas separadas. Em seguida, o método faz a combinação dois a dois de todos os pontos (clientes e centro de distribuição) e calcula a lista de economias que contém os ganhos de todos os pares. É feita a ordenação dessa lista em ordem decrescente e a partir dessa ordenação os pontos são analisados. A

união de pares é realizada considerando as restrições do problema e caso o ganho seja maior que zero, a união é considerada e colocada na lista de economias.

3.1.2 Algoritmo Simulated Annealing

Em (HARMANANI et al., 2011), o método heurístico *Simulated Annealing* foi adaptado para o problema de roteamento de veículos. Nesse método, a melhor solução é procurada na vizinhança da solução atual, porém, a diferença com outros métodos que utilizam a mesma abordagem é que as soluções vizinhas são geradas a partir da escolha aleatória de uma operação entre um conjunto de operações determinístas. Se essa solução gerada for melhor, ela sempre é aceita como a nova solução atual, mas se a solução é pior, essa só é aceita com certa probabilidade. Esta probabilidade é determinada por uma temperatura que é gradualmente diminuída. Ao reduzir a temperatura, a seleção se torna cada vez mais seletiva ao aceitar uma nova solução.

Para o VRP, cada possível solução de rotas é chamada de configuração. O algoritmo começa por selecionar uma configuração inicial e em cada iteração do algoritmo é gerada uma nova configuração vizinha utilizando algumas operações de transformação, entre elas as transformações chamadas de *mover* e *substituir a média mais alta*.

3.1.3 Algoritmo baseado em Centroide

Em (SHIN; HAN, 2012), é proposto um algoritmo heurístico baseado em centroide para resolver o VRP. O algoritmo consiste em três fases: construção de clusters, ajuste de clusters e estabelecimento de rota.

A primeira fase tem como objetivo separar os clientes em grupos pela distância. A segunda ajusta os agrupamentos e a última fase busca melhorar as rotas resolvendo o problema caixeiro-viajante (Traveling Salesman Problem – TSP) para os grupos estabelecidos.

Na primeira fase, são construídos grupos (clusters) de clientes. O grupo é construído selecionando o cliente mais distante do depósito como semente do grupo. Após isso, são selecionados os clientes mais próximos do centroide do cluster, sempre atualizando a localização do centroide ao adicionar um novo cliente. Esse processo continua enquanto a

capacidade do veículo não for ultrapassada. Quando isso acontece, a construção do grupo é finalizada, uma nova semente é selecionada e o processo de construção de um novo grupo começa novamente até que todos os clientes estejam em um grupo.

A segunda fase, ajusta os grupos formados com auxílio do centro geométrico dos grupos. Essa etapa verifica se um cliente está mais próximo do centroide de um grupo vizinho do que o centroide de seu grupo atual. Caso isso aconteça e a capacidade do veículo do grupo vizinho não seja violada, o cliente é movido para o grupo vizinho e os centroides são recalculados.

Na última fase, as rotas são estabelecidas para o atendimento dos clientes de cada grupo. Para isso, a heurística apresentada em (LIN; KERNIGHAN, 1973) é usada para resolver o TSP.

3.1.4 Algoritmo Monte Carlo Savings

O algoritmo Monte Carlo Savings é baseado no algoritmo Clarke & Wright e usa técnicas de Simulação de Monte Carlo. Diferente dos outros trabalhos que unem essas duas técnicas, o estudo apresentado em (OLIVEIRA; DELGADO; MOREIRA, 2016) utiliza as Simulações de Monte Carlo para variar a ordem em que as arestas da lista de economias são percorridas. Dessa forma são alteradas as soluções finais e é permitido que diferentes rotas sejam geradas em cada simulação.

O algoritmo executa 2000 simulações, sendo que em cada uma delas é gerada uma lista de economias. Para cada lista gerada, o algoritmo procede como o método de Clarke & Wright. A lista é percorrida de forma decrescente fundindo as rotas viáveis. Para permitir a variação nas listas de economias, a fórmula de cálculo de economia recebe um componente aleatório, o qual é usado para bonificar ou penalizar a economia gerada para a aresta.

3.2 VRP com janela de tempo

O problema de roteamento de veículos com janela de tempo (em inglês Vehicle Routing Problem with Time Window - VRPTW) é uma extensão do problema VRP, em que, para aproximar o problema a casos do mundo real, são consideradas janelas de tempo de atendimento do cliente. Assim, o início do serviço (entrega) a um cliente deve começar

dentro do prazo definido por uma janela de tempo que inclui o horário de início e fim (TOTH; VIGO, 2002).

Por se tratar de um problema muito bem conhecido no ambiente de distribuição e prestação de serviços, existem diversos métodos que são utilizados para encontrar uma solução para o VRPTW, entre eles métodos exatos (FISHER; JÖRNSTEN; MADSEN, 1997; KOHL; MADSEN, 1997), heurísticos (BAKER; SCHAFFER, 1986; SCHULZE; FAHLE, 1999) e meta-heurísticos (CHIANG; RUSSELL, 1996; RUSSELL, 1995). Esses métodos são compilados em (EL-SHERBENY, 2010).

Um dos primeiros trabalhos a propor uma solução para o VRPTW é (SOLOMON, 1987) em que é proposta uma extensão do já estabelecido método de Clarke & Wright para o problema de janela de tempo. O método proposto é similar ao detalhado na Seção 3.1.1. Ambos apresentam a mesma função de economia e procedimento para construção da solução. Porém, a cada validação de uma economia os autores acrescentaram a validação das restrições referentes ao problema de janela de tempo. Além disso, foi observado que dessa forma o método poderia juntar dois clientes muito próximos em distância, mas distantes no tempo. Assim, essa junção acaba gerando um grande período de espera para início do atendimento dos clientes. Dessa forma, os autores adicionaram um limite de tempo de espera ao considerar um par de clientes na construção da solução.

Outro trabalho que visa solucionar o VRPTW é o de (TAN et al., 2001), uma adaptação do tradicional método Simulated Annealing. Esse trabalho é próximo ao método descrito na seção 3.1.2, no qual uma solução representa uma configuração de rotas para atender as entregas dos clientes. Porém, o modelo matemático do método de (TAN et al., 2001) considera as restrições de janelas de tempo para cada cliente. A operação de movimento para localizar as soluções vizinhas da solução atual e que considera essas janelas aplicada nesse método é chamada de 2-interchange LSD(GB).

4 Problema de roteamento e estoque

Neste capítulo são definidos os problemas de roteamento e estoque (IRP), determinístico e estocástico. São apresentadas a classificação das variações estruturais do IRP e a classificação dos principais algoritmos identificados durante a revisão da literatura.

4.1 Problema básico de IRP

Uma modelagem padrão é apresentada para o problema IRP em (COELHO; CORDEAU; LAPORTE, 2013). Uma instância do problema IRP é definida como um grafo $G = (S, A)$, sendo que $S = \{0, \cdots, n\}$ é o conjunto de vértices e $A = \{(i, j) : i, j \in S, i \neq j\}$ são os arcos. O vértice 0 representa o fornecedor e os vértices $\zeta = S \setminus \{0\}$ são os clientes. Um custo de rota Θ_{ij} é atribuída a cada arco $(i, j) \in A$. O horizonte de planejamento é definido com um tamanho T e cada período $h \in H = \{1, \cdots, T\}$ e $H^+ = H \cup \{0\}$. Cada cliente tem uma capacidade de estoque C_j e tanto fornecedores quanto clientes têm um custo de estocagem η_{jh} por período, em que $(j \in \zeta, h \in H)$.

A quantidade de produto que será entregue ao fornecedor no tempo h é R_{rh}, sendo essa variável resultante da política de inventário do fornecedor. Os níveis de estoque do fornecedor e dos clientes são representados no final do período h por I_{0h} e I_{jh} $(j \in \zeta, h \in H)$, respectivamente. No início do horizonte de planejamento, o nível de estoque do fornecedor e dos clientes são definidos como (I_{00} *e* I_{j0} *em que* $j \in S$). A demanda no modelo básico do IRP é definida como d_{jh} para cada cliente j no período h. Os veículos são representados pelo conjunto $V = \{1, \cdots, K\}$ com capacidade de transporte k_v.

O objetivo do problema é reduzir o custo da distribuição, planejando o reabastecimento da demanda de cada cliente. As restrições do problema são (COELHO; CORDEAU; LAPORTE, 2013):

- O nível de estoque em cada cliente nunca pode exceder a sua capacidade máxima;
- Os níveis de estoque não podem ser negativos;
- Os veículos do fornecedor podem executar um percurso por período, sendo que eles se iniciam e terminam no fornecedor;
- A capacidade de transporte do veículo não pode ser ultrapassada.

Tabela 1 – Variações estruturais do IRP

Critério	Possíveis Opções		
Horizonte de tempo	Finito	Infinito	
Estrutura distribuidor-cliente	Um-para-Um	Um-para-Muitos	Muitos-para-Muitos
Roteirização	Direta	Múltipla	Continua
Política de estoque	Nível Máximo	Nível Order-up-to	
Decisão de estoque	Venda perdida	Back-order	Não negativa
Composição da frota	Homogênea	Heterogênea	
Tamanho da frota	Única	Múltipla	Sem restrição

Fonte: Adaptado de (COELHO; CORDEAU; LAPORTE, 2013)

A solução proposta deve determinar o período em que o cliente será atendido e quanto do produto será entregue nessa visita. Essa quantidade de produtos a ser entregue em um período especifico é descrita como $(q_{jh}, j \in S, h \in H)$, sendo que as rotas são propostas visando minimizar os custos totais da utilização do veículo.

4.2 Categorização de IRPs

Os IRPs podem ser classificados usando diferentes critérios que são definidas como as variações estruturais do problema, sendo elas horizonte de tempo, estrutura, roteirização, política de estoque, decisão de estoque, composição da frota e tamanho da frota (COELHO; CORDEAU; LAPORTE, 2013). As possíveis opções para cada critério são apresentados na Tabela 1.

O horizonte de tempo indica o período de planejamento, podendo ser finito ou infinito. O número de clientes e dos centros de distribuição de um fornecedor podem variar, assim a estrutura distribuidor-cliente pode ser Um-para-Um quando só existe um centro de distribuição atendendo um cliente, Um-para-Muitos quando um centro de distribuição atende diversos clientes e Muitos-para-Muitos no qual vários centros de distribuição atendem vários clientes. A roteirização pode ser direta quando só existe um cliente por rota, múltipla quando existem diversos clientes em uma mesma rota ou contínuo quando não existe um centro de distribuição central, como em aplicações marítimas.

As políticas de estoque definem regras para reabastecer os clientes. Na política de nível máximo, o nível de reabastecimento é flexível, mas limitado pela capacidade disponível a cada cliente. Na política de Nível Order-up-to, sempre que um cliente é visitado, a quantidade entregue preenche a sua capacidade de estoque. A decisão de estoque indica

como seu gerenciamento é aplicado. Uma das opções é considerar que o estoque nunca pode ser negativo. A outra opção é permitir que o estoque alcance valores negativos, então pode acontecer back-order ou venda perdida. No primeiro caso, a demanda correspondente não atendida vai ser entregue posteriormente, e no segundo, a demanda não atendida é considerada como venda perdida.

A primeira classificação com relação à frota é sobre a sua composição, podendo ser composta de veículos homogêneos, i.e., que têm uma mesma capacidade, ou veículos heterogêneos, que têm capacidades diferentes. Também temos a classificação pelo tamanho da frota, sendo que essa pode ter apenas um veículo, múltiplos veículos ou pode não existir restrição do tamanho da frota.

Os IRPs também podem ser classificados com relação ao tempo de quando a informação da demanda se torna conhecida. Se essa informação está totalmente disponível durante a tomada de decisão, i.e., no início do horizonte de planejamento, o problema IRP é determinístico. Caso somente a informação da distribuição de probabilidade da demanda seja conhecida, então esse IRP é estocástico. Esse problema é conhecido como Problema de Roteamento e Estoque com Demanda Estocástica que considera que a demanda d_{jh} dos clientes pode não ser conhecida completamente (COELHO; CORDEAU; LAPORTE, 2013).

Dessa forma, a estocasticidade da demanda implica que pode acontecer escassez. Para desencorajar isso, os trabalhos que visam solucionar esse problema, costumam acrescentar penalidades em seus modelos quando a solução tende para a escassez. As técnicas propostas para SIRPs têm o mesmo objetivo que sua versão determinística, porém com a dificuldade de atender a estocasticidade e os futuros valores desconhecidos da demanda.

4.3 IRP determinístico

Existem vários modelos e soluções propostos pelos especialistas para o IRP com demanda determinística, a maioria foi resumido por Bertazzi, Savelsbergh e Speranza (2008). Existem algumas soluções que usam programação inteira como (CHIEN; BALAKRISHNAN; WONG, 1989) e (CAMPBELL; CLARKE; SAVELSBERGH, 2002) ou utilizam processo de decisão markoviano resolvido com programação dinamica como (CAMPBELL et al., 1998) e (CAMPBELL; CLARKE; SAVELSBERGH, 2002).

Porém, a maioria dos trabalhos utilizam abordagens heurísticas ou meta-heurísticas, como por exemplo (ANILY; FEDERGRUEN, 1990), (ANILY; FEDERGRUEN, 1993), (BRAMEL; SIMCHI-LEVI, 1995), (CHAN; FEDERGRUEN; SIMCHI-LEVI, 1998), (BERTAZZI; PALETTA; SPERANZA, 2002) e (COUSINEAU-OUIMET, 2002). Outros autores sugerem soluções sequênciais como, por exemplo, (CAMPBELL; SAVELSBERGH, 2004) que apresentam uma solução de duas fases incluindo o suporte para o processo de busca adaptativa randomizada gulosa (em inglês Greedy Randomized Adaptive Search Procedure – GRASP).

4.4 IRP com demanda estocástica

Em (COELHO; CORDEAU; LAPORTE, 2013) foram identificados três grupos de solução para o SIRP que são: (i) algoritmos heurísticos que geralmente usam políticas de entrega predefinidas; (ii) algoritmos baseados em Processos de Decisão Markovianos e que usam programação dinâmica; e (iii) algoritmos que utilizam otimização robusta. A Tabela 2 apresenta os artigos citados em (COELHO; CORDEAU; LAPORTE, 2013) e mais alguns encontrados durante a revisão da literatura. Esses artigos foram organizados conforme a classificação apresentada em (COELHO; CORDEAU; LAPORTE, 2013).

4.4.1 Algoritmos heurísticos

Um dos primeiros algoritmos para resolver SIRP foi proposto por Federgruen e Zipkin (1984a). Esse algoritmo estende a solução heurística do VRP proposta por Fisher e Jaikumar (1981) adicionando os custos de escassez, transporte, estocagem e a demanda aleatória. O problema considerado é de um único período em que os recursos escassos que estão estocados em um depósito devem ser distribuídos entre múltiplos clientes. Eles propõem diversos exemplos de aplicações, incluindo entregas de combustível a postos de gasolina, reabastecimento periódico de tanques de gás em clientes, alocação coordenada e fornecimento para vários locais de um produto perecível, como por exemplo reabastecimento de sangue em hospitais. A solução proposta inclui um modelo matemático, uma heurística de trocas e um algoritmo exato para resolver instâncias geradas aleatoriamente com 75 clientes.

Tabela 2 – Classificação das variações estruturais dos artigos sobre SIRP

Referência	Tempo de Horizonte		Estrutura			Roteirização			Política de Estoque		Decisão de Estoque		Composição da Frota		Tamanho da Frota		
	Finito	Infinito	Um-para-Um	Um-para-Muitos	Muitos-para-Muitos	Direto	Multiplo	Contínuo	Nivel Maximo	Order-up-to	Venda Perdida	Backlogging	Homogênea	Heterogênea	Único	Múltiplo	Sem Restrição
Algoritmos Heurísticos																	
(FEDERGRUEN; ZIPKIN, 1984a)	x			x			x		x		x			x		x	
(FEDERGRUEN; ZIPKIN, 1984b)		x		x			x		x			x		x			x
(FEDERGRUEN; PRASTACOS; ZIPKIN, 1986)	x			x			x		x		x			x		x	
(GOLDEN; ASSAD; DAHL, 1984)	x			x			x		x		x		x			x	
(TRUDEAU; DROR, 1992)	x		x				x		x		x		x			x	
(MINKOFF, 1993)		x		x			x		x		x		x				x
(BARNES-SCHUSTER; BASSOK, 1997)		x		x		x				x		x	x		x		
(BARD et al., 1998)	x				x		x		x		x		x			x	
(QU; BOOKBINDER; IYOGUN, 1999)		x		x			x		x		x		x		x		
(REIMAN; RUBIO; WEIN, 1999)		x		x		x			x			x	x		x		
(BERMAN; LARSON, 2001)		x		x			x		x		x		x		x		
(JAILLET et al., 2002)		x		x			x			x		x	x				x
(GAUR; FISHER, 2004)	x			x			x		x			x		x		x	
(BINGLEI; SHI; JIAN, 2005)	x			x			x		x		x		x				x
(CUSTÓDIO; OLIVEIRA, 2006)		x		x			x		x		x		x			x	
(JARUGUMILLI; GRASMAN; RAMAKRISHNAN, 2006)	x			x			x		x			x	x			x	
(JARUGUMILLI; GRASMAN, 2006)	x			x			x		x			x	x			x	
(LOU; WU; XIAO, 2009)	x				x		x		x		x		x			x	
(HVATTUM; LØKKETANGEN, 2009)		x		x			x		x		x		x			x	
(HVATTUM; LØKKETANGEN; LAPORTE, 2009)		x		x			x		x		x		x			x	
(HUANG; LIN, 2010)		x		x		x				x	x		x			x	
(GEIGER; SEVAUX, 2011)	x			x			x		x	x	x		x				x
(CÁCERES-CRUZ et al., 2012)	x			x			x		x		x		x				x
(SHUKLA; TIWARI; CEGLAREK, 2013)		x		x			x		x		x		x			x	
(NOLZ; ABSI; FEILLET, 2014)	x			x			x		x		x		x			x	
(JUAN et al., 2014)	x			x			x		x		x		x				x
Algoritmos que usam programação dinâmica para Processo de Decisão Markovianos																	
(CAMPBELL et al., 1998)		x		x			x		x		x		x				x
(KLEYWEGT; NORI; SAVELSBERGH, 2002)		x		x		x			x		x		x				x
(KLEYWEGT; NORI; SAVELSBERGH, 2004)		x		x			x		x		x		x			x	
(ADELMAN, 2004)		x		x			x		x		x		x			x	
Algoritmos baseados em Otimização Robusta																	
(YU; CHU; CHEN, 2006)	x			x			x		x		x		x				x
(AGHEZZAF, 2008)		x		x			x		x		x		x			x	
(SOLYALI; CORDEAU; LAPORTE, 2012)	x			x			x		x			x	x		x		
(BERTAZZI et al., 2013)	x			x			x			x		x	x		x		
(RAHIM et al., 2014)	x			x			x		x		x		x				x

Fonte: Adaptado de (COELHO; CORDEAU; LAPORTE, 2013)

Em (FEDERGRUEN; ZIPKIN, 1984b) foi provado que a convergência para boas soluções pode ser rápida assumindo hipóteses para os padrões de políticas de distribuição. Foram realizados testes com 4 conjuntos de 192 instâncias variando a média e a variância da demanda de distribuição. Federgruen, Prastacos e Zipkin (1986) estendem os trabalhos anteriores para considerar múltiplos produtos perecíveis no modelo proposto.

Já em (GOLDEN; ASSAD; DAHL, 1984), os clientes são classificados considerando uma escala de urgência para atender a demanda estocástica. Essa classificação é usada para definir qual cliente deverá ser atendido e após esse processo, o problema de roteamento é evidenciado e resolvido por um método de TSP (em inglês Traveling Salesman Problem – TSP).

O SIRP considerando escassez e falha durante a operação das rotas é analisado por Trudeau e Dror (1992). Esse trabalho apresenta um novo procedimento para selecionar qual cliente será atendido em cada período. Após a seleção é analisado o custo esperado com escassez do produto e possíveis falhas durante a operação da rota.

Minkoff (1993) é um dos poucos que trabalha com frota sem restrição, propondo uma abordagem heurística em que a função objetivo é simplificada, dividindo e criando uma função objetivo para cada cliente e finalmente o problema é resolvido de forma heurística considerando a soma das funções objetivo.

Barnes-Schuster e Bassok (1997) sugerem um cenário do problema considerando um horizonte infinito em que a demanda é continua e estocástica. Nesse trabalho, foi concluído que o transporte direto, aquele em que um veículo abastece um cliente e retorna para o depósito, pode ser uma boa estratégia quando a demanda do cliente se aproxima da capacidade máxima do veículo.

Em (BARD et al., 1998) é considerado um SIRP com instalações satélites (depósitos geograficamente espalhados por toda a área de atuação) que permitem que o veículo reabasteça os produtos durante a execução. Foi utilizada uma versão da heurística de economias de Clarke & Wright (CWS) (CLARKE; WRIGHT, 1964) para resolver o subproblema de roteamento para 500 clientes em 2 horas. Com isso, é mostrado que essa heurística supera outros algoritmos, incluindo GRASP.

Qu, Bookbinder e Iyogun (1999) desenvolve uma política de entrega periódica para um SIRP com múltiplos produtos.

Reiman, Rubio e Wein (1999) consideram um SIRP com um veículo cobrindo uma região com diversos clientes. Três estratégias são criadas e seus resultados comparados.

A primeira estratégia se baseia em transporte direto, a segunda tem como premissa a utilização de uma rota pré-determinada solucionada por um algoritmo de TSP e a terceira realiza escolhas entre o TSP e a opção de rotas com transporte direto. Os autores concluem que a primeira estratégia teve melhor desempenho na maioria dos casos.

A heurística de Berman e Larson (2001) é auxiliada por programação dinâmica para resolver o caso quando a distribuição de probabilidade é conhecida, ajustando a quantidade de produto entregue em cada cliente.

Jaillet et al. (2002) defendem a utilização de horizontes de planejamento curtos quando se lida com o SIRP pois segundo eles, a taxa de consumo de um cliente é difícil de prever com certeza e só pode ser representado no melhor dos casos por uma variável aleatória com distribuição de probabilidade conhecida. Além disso, defendem que planejar todo o esquema de distribuição anual com antecedência poderia não ser confiável e propenso a muitas adaptações. Os autores assumem que o depósito não pode rastrear o nível do estoque de seus clientes, sendo que esse só fica visível após a visita do veículo. Assim, nesse trabalho são propostas diversas políticas de reabastecimento sobre um único período.

Gaur e Fisher (2004) consideram um SIRP com frota heterogênea. Para simplificar o problema, os clientes foram separados em diversas regiões disjuntas. Originalmente, foram consideradas partições de no máximo 2 clientes e após todas as partições serem criadas, uma heurística foi utilizada para melhorar a combinação. Porém, eles não consideram o custo de estoque pois o custo considerado relevante é apenas o custo de roteirização.

O trabalho proposto em (BINGLEI; SHI; JIAN, 2005) considera um SIRP com veículos homogêneos e apresenta um método de aproximação para lidar com a demanda estocástica. Após a aproximação é possível calcular o período e a quantidade de distribuição para cada cliente, reduzindo o problema a um VRP para cada período.

Um caso real de SIRP com múltiplos produtos é estudado em (CUSTÓDIO; OLIVEIRA, 2006). Esse trabalho considera três tipos de políticas: política de particionamento fixo, política de transporte direto e política de proporções de inteiros. Na política de particionamento fixo, o conjunto de produtos é dividido em grupos que sejam reabastecidos independentemente. Na política de transporte direto, cada veículo atende apenas um cliente. Por fim, na política de proporções de inteiros, todos os produtos são repostos constantemente em um intervalo de tempo. Após discutir essas estratégias é proposta uma heurística como solução.

Uma modificação do algoritmo A^* é desenvolvida em (JARUGUMILLI; GRASMAN; RAMAKRISHNAN, 2006) para solucionar o SIRP com um único veículo. Esse trabalho é estendido em (JARUGUMILLI; GRASMAN, 2006) ao incorporar informação em tempo real.

Um novo método para solucionar o SIRP é proposto em (LOU; WU; XIAO, 2009) considerando múltiplos depósitos. Após a definição do modelo que considera diversos custos (por exemplo, inventário ativo e estoque de segurança) são utilizados algoritmos auxiliares para tratar a convergência do método tradicional de decomposição e coordenação. Esse método consiste em decompor o problema principal em subproblemas e coordenar a distribuição ótima entre um conjunto de clientes. Os valores da distribuição do conjunto de clientes são criados por um algoritmo genético, em seguida a busca Tabu é usada para lidar com os custos de roteamento esperados pela amostragem de Monte Carlo.

Hvattum e Løkketangen (2009) consideram um SIRP com horizonte infinito. Eles modelam o problema como um processos de decisão markoviano, porém não utilizam técnicas de programação dinâmica para encontrar uma solução. A solução do SIRP é complexa por considerar um horizonte infinito e um número muito grande de possíveis demandas para os clientes. Para lidar com esses dois problemas, os autores utilizam árvores de cenários para aproximar a solução considerando apenas um horizonte finito e limitando a quantidade de possíveis demandas. A heurística adotada para solução desse modelo é baseada no algoritmo GRASP.

Em (HVATTUM; LØKKETANGEN; LAPORTE, 2009), é proposta uma extensão do trabalho de Hvattum e Løkketangen (2009), utilizando a modelagem de MDP com árvore de decisão. Uma heurística baseada no algoritmo de restrição progressiva proposto por Rockafellar e Wets (1991) é desenvolvida para resolver as mesmas instâncias de teste que o trabalho anterior, porém essa técnica demonstrou ser menos robusta que sua antecessora.

Já em (HUANG; LIN, 2010), é proposta uma solução por meio de um algoritmo de otimização de colônia de formigas, incorporando as questões de escassez através da atração dos valores do feromônio em cada nó.

Em (GEIGER; SEVAUX, 2011), são propostas várias políticas baseadas em frequência de entrega para cada cliente. Com o objetivo de identificar soluções Pareto-ótimas, são considerados aspectos opostos na função objetivo de seu modelo. Se um cliente

é visitado frequentemente, o custo de armazenamento é baixo mas o custo de roteamento é alto, o caso contrário também é considerado.

Cáceres-Cruz et al. (2012) dividem o SIRP em dois subproblemas: o problema de roteamento e o problema de gerenciamento de estoque. Nesse trabalho é proposta uma heurística hibrida que inclui simulação de Monte-Carlo (em inglês Monte Carlo Simulation – MCS) e um método eficiente para solução de roteamento de veículo com capacidade, como o CWS. O subproblema de gerenciamento de estoque irá lidar com as variáveis estocásticas através da MCS devido à eficiência desse método em trabalhar com problemas estocásticos. Assim, o MCS é utilizado para obter estimativas do estoque de cada cliente. Os resultados mostram que o MCS não chega perto da solução ótima, mas resulta em uma solução quase ótima. Além disso, é demonstrado que a principal vantagem desse método é a possibilidade de ter políticas de abastecimento personalizadas para cada cliente.

Shukla, Tiwari e Ceglarek (2013) apresentam um portfólio de algoritmos baseados em algoritmos evolutivos para solucionar o SIRP. O portfólio consiste de variações do algoritmo genético: algoritmo memético, algoritmo genético com diferenciação cromossômica, algoritmo de idade genética e algoritmo genético específico do gênero.

O método apresentado por Nolz, Absi e Feillet (2014) começa por uma solução inicial e gradualmente a aprimora usando mecanismos que irão desconstruir e reparar a solução. Em cada etapa de desconstrução e reparação define-se um peso que é a representação do seu desempenho durante o processamento.

Por fim, Juan et al. (2014) consideram um SIRP de um único período em que diversas políticas de reabastecimento personalizadas para cada cliente são consideradas. O trabalho utiliza uma abordagem de simulação e otimização para qualquer modelo de distribuição de probabilidade da demanda. O algoritmo proposto, chamado *Simheuristic*, apresenta múltiplos-começos randômicos, o que permite a utilização de processamentos paralelos. Nesse trabalho, foram utilizadas mais de 405 instâncias de testes geradas aleatoriamente.

4.4.2 Algoritmos baseados em processo de decisão markovianos

Campbell et al. (1998) foram os primeiros a abordar o SIRP como um processo de decisão markoviano. Para simplificar o modelo, somente o custo de transporte e de escassez foram considerados. O fornecedor tem que decidir quanto irá entregar ao cliente,

como organizar a entrega para os clientes em rotas e associar cada rota a um veículo, sendo que no começo de cada período do horizonte de planejamento, o fornecedor conhece o nível de estoque dos seus clientes.

Kleywegt, Nori e Savelsbergh (2002) consideram um SIRP de horizonte infinito e trabalham somente com transporte direto. O problema é modelado como um processo de decisão markoviano de tempo discreto e são sugeridos diversos métodos de aproximação como solução. Kleywegt, Nori e Savelsbergh (2004) estendem o trabalho anterior considerando rotas de no máximo três clientes e inserem vários recursos para melhorar a segurança e reduzir a contaminação entre produtos durante o transporte e armazenamento. São resolvidas instâncias com até 20 clientes em um tempo computacional razoável.

Já Adelman (2004) não restringe o número de clientes que podem ser colocados em uma mesma rota, porém considera nessa alocação a duração máxima da rota e a capacidade do veículo. Foi utilizada programação inteira mista e processos de Markov para resolver o SIRP de horizonte infinito. A maior instância gerada aleatoriamente utilizada no experimento contém 40 clientes.

4.4.3 Algoritmos baseados em otimização robusta

Yu, Chu e Chen (2006) consideram o SIRP com possibilidade de entrega dividida, em que um único cliente pode ser atendido por dois veículos distintos. A sua solução utiliza relaxação Lagrangeana para decompor o modelo em um subproblema de estoque e outro de roteirização, em seguida uma solução quase ótima é construída. Foram resolvidas instâncias de 100 clientes. Porém, o seu modelo é mais simples (com menos variáveis) do que trabalhos mais recentes na literatura. Além disso, as soluções encontradas podem apresentar alguns problemas, por exemplo, permitir que um veículo trafegue vazio entre clientes após as entregas no lugar de voltar para o depósito.

Aghezzaf (2008) considera uma distribuição normal de probabilidade da demanda dos clientes e tempo de viagem. Essa solução utiliza otimização robusta combinada com uma formulação de programação inteira mista não linear para determinar um plano de distribuição. Para definir os parâmetros do problema como nível de estoque de segurança e os ciclos de reposição do estoque, foi utilizado simulação de Monte Carlo.

Solyali, Cordeau e Laporte (2012) consideram um SIRP em que um fornecedor realiza a distribuição de um único produto para n clientes usando veículos de capacidades homogêneas sobre um horizonte finito. A probabilidade de distribuição da demanda de cada cliente é desconhecida, porém o intervalo de variação da distribuição é conhecido. São permitidos atrasos na entrega e cada unidade de produto que não satisfez a demanda esperada tem um custo. A função objetivo é o total dos custos fixos do veículo, transporte, estocagem e escassez. São resolvidas instâncias com mais de trinta clientes e com um horizonte contendo sete períodos.

Bertazzi et al. (2013) consideram SIRPs com falta de estoque e com um horizonte finito. Nesse trabalho é usada uma política order-up-to, i.e., uma política em que a quantidade enviada para cada cliente é tal que seu nível de estoque atinge o nível máximo sempre que o cliente é abastecido. Os autores apresentam um modelo de otimização robusta e propõem um algoritmo de rollout híbrido. Afim de validar a sua abordagem, eles usam um conjunto de instâncias geradas aleatoriamente com até 50 clientes e 6 períodos.

Em (RAHIM et al., 2014) é utilizado relaxação Lagrangeana para resolver um SIRP de horizonte finito. Com o auxílio do algoritmo de subgradiente, são atualizados o multiplicador lagrangeano e encontrados os limites inferiores e superiores da solução. Esse algoritmo é descrito de forma detalhada no Capítulo 5.

4.5 IRP determinístico com janela de tempo

O problema de janela de tempo é inicialmente apresentado como uma extensão do problema VRP. Conforme definido em (CORDEAU et al., 2000), cada cliente j tem uma janela de tempo $[b_j, e_j](j \in \zeta)$, sendo que o atendimento desse cliente j deve iniciar dentro deste intervalo de tempo.

Em (SOLOMON; DESROSIERS, 1988), são apresentadas duas categorias de restrições de janelas de tempo: fraca e forte. No caso da janela de tempo forte, não pode haver violação da janela de tempo, garantindo o atendimento combinado entre fornecedor e cliente. Considerando $t_j, j \in \zeta$ o horário de chegada e início de atendimento do cliente, a restrição desse tipo de janela de tempo pode ser definida como:

$$b_j \leq t_j \leq e_j, \quad \forall j \in \zeta$$

A restrição fraca adiciona a função objetivo do modelo uma penalidade $P_j(t_j), j \in \zeta$ pelo não atendimento da janela, permitindo que ocorram atrasos ou esperas; e dessa forma não restringe o atendimento no horário especificado. Essa penalidade pode ser definida como:

$$P_j(t_j) = \left\{ \begin{array}{cc} P_b(b_j - t_j), & se\ t_j < b_j \\ 0 & se\ b_j \leq t_j \leq e_j \\ P_e(t_j - e_j), & se\ e_j < t_j \end{array} \right\}, j \in \zeta$$

Sendo que $P_b(b_j - t_j)$ indica a penalidade por violar a janela de início do atendimento e $P_e(t_j - e_j)$ a penalidade por chegar após o fim da janela de atendimento.

Ao combinar o IRP com o problema de janela de tempo, temos que em cada período de planejamento existe uma janela de tempo para cada cliente, esse problema é conhecido como Problema de Roteamento e Inventário com Janela de Tempo (em inglês Inventory Routing Problem with Time Window – IRPTW).

A maioria dos estudos encontrados utilizam somente uma abordagem heurística para resolver o problema com exceção de (LI et al., 2014) e (IASSINOVSKAIA; LIMBOURG; RIANE, 2017). Li et al. (2014) além de apresentar um método heurístico de busca Tabu, apresentam também um método de relaxação lagrangeana para resolver o modelo de otimização robusta. Já Iassinovskaia, Limbourg e Riane (2017) propõem um algoritmo branch-and-cut de programação inteira mista.

As características estruturais dos estudos encontrados estão classificadas nas Tabelas 3 e 4.

Tabela 3 – Classificação das variações estruturais dos IRPTW

Referência	Tempo de Horizonte		Estrutura			Roteirização			Política de Estoque		Decisão de Estoque			Composição da Frota		Tamanho da Frota		
	Finito	Infinito	Um-para-Um	Um-para-Muitos	Muitos-para-Muitos	Direto	Múltiplo	Contínuo	Nível Máximo	Order-up-to	Venda Perdida	Backlogging	Não Negativo	Homogênea	Heterogênea	Único	Múltiplo	Sem Restrição
(CAMPBELL; SAVELSBERG, 2004)	X			X			X			X	X			X		X		
(RUSDIANSYAH; TSAO, 2005)	X			X			X		X				X	X			X	
(LIU; LEE, 2011)	X			X			X		X			X		X			X	
(ARCHETTI; DOERNER; TRICOIRE, 2013)	X			X			X		X		X			X				X
(JIA et al., 2014)	X			X			X		X				X	X				X
(LI et al., 2014)	X			X			X			X			X		X		X	
(XIAO; RAO, 2016)	X				X		X		X				X	X			X	
(IASSINOVSKAIA; LIMBOURG; RIANE, 2017)	X			X			X		X				X	X			X	

Tabela 4 – Classificação dos estudos pelo tipo de restrição de janela de tempo

Referência	Janela de Tempo	
	Forte	Fraco
(CAMPBELL; SAVELSBERG, 2004)	X	
(RUSDIANSYAH; TSAO, 2005)	X	
(LIU; LEE, 2011)		X
(ARCHETTI; DOERNER; TRICOIRE, 2013)	X	
(JIA et al., 2014)		X
(LI et al., 2014)	X	
(XIAO; RAO, 2016)	X	
(IASSINOVSKAIA; LIMBOURG; RIANE, 2017)	X	

4.5.1 Algoritmos heurísticos

Em (CAMPBELL; SAVELSBERG, 2004) é apresentado um algoritmo de tempo linear para resolver um problema chamado de *otimização do volume de entrega* que por sua vez pode ser usado em abordagens heurísticas para resolver o problema de IRPTW. O problema de *otimização do volume de entrega* consiste em determinar o cronograma de entrega em uma rota considerando que: (i) existem janelas de tempo de entrega nos clientes; (ii) a quantidade máxima que pode ser entregue em um cliente depende do tempo de entrega; e (iii) as entregas não são instantâneas e dependem do tamanho da entrega. No algoritmo de otimização são usados os perfis de volume de demanda de cada cliente definidos com base no consumo da demanda de cada um. A partir da identificação do cliente que tem o maior pico de demanda durante os períodos de planejamento, uma política de atendimento é desenvolvida assumindo que o caso ótimo seria entregar primeiro para aquele cliente que tem o maior volume de demanda, no período identificado, e em seguida entregar o máximo possível para os demais clientes, seguindo as restrições do problema IRPTW.

Em (RUSDIANSYAH; TSAO, 2005) um modelo para IRPTW é construído com base no problema conhecido como *roteamento periódico de veículos com janelas de tempo*. Além disso, uma heurística de duas fases é desenvolvida. A primeira fase, chamada de inicialização, tem como objetivo encontrar uma frequência viável de visita para cada cliente. Para isso assume-se inicialmente que cada cliente tem a menor frequência de visita possível, em seguida é escolhido entre os clientes, um número determinado de sementes para inicializar cada rota de operação. A próxima etapa da inicialização é aplicar a heurística chamada de *inserção mais barata*, que busca o custo de inserção mais baixo para atribuir os clientes não selecionados como sementes. E por último, é aplicado o algoritmo de

otimização de frequência de visita, no qual gradualmente é aumentada a frequência de visita de cada cliente até a maior frequência permitida. Na segunda fase, são aplicadas heurísticas de troca que podem melhorar a solução atual. Basicamente duas heurísticas são utilizadas:

- *Troca de combinação visita-dia*: constrói vizinhos de uma solução inicial alterando combinações de visitas diárias de cada cliente para reduzir o custo de viagem, e em seguida, a busca Tabu é utilizada para encontrar o melhor vizinho.
- *Troca de rotas*: realiza três operações distintas considerando o custo de inserção mínimo viável. A primeira transfere um único cliente de sua posição na rota para outra posição na mesma rota. A segunda transfere um cliente de sua rota para outra posição em uma rota diferente no mesmo dia. E por fim, a terceira aplica o método de troca 2-opt, que é um algoritmo de busca local simples, sua versão original foi proposta em (CROES, 1958). A principal ideia desse algoritmo é identificar possíveis cruzamentos no percurso de uma rota e reordenar sua sequencia para que esses cruzamentos sejam desfeitos.

Liu e Lee (2011) apresentam um modelo matemático para IRPTW que minimiza o custo de transporte, o custo de violação da janela de tempo e o custo de inventário. Este estudo também apresenta uma heurística de duas fases semelhante a (RUSDIANSYAH; TSAO, 2005). A primeira fase encontra a solução viável inicial e a partir dessa solução inicial a segunda fase propõe encontrar melhores soluções aplicando o algoritmo *Variable Neighborhood Tabu Search* (PÉREZ; MORENO-VEGA; MARTIN, 2003). Os vizinhos são gerados por duas estratégias:

- A partir da solução inicial é aplicado um procedimento de melhoria da roteirização dos veículos, selecionando um número de clientes de diferentes rotas e excluindo estes das rotas originais, em seguida estes são inseridos em outras rotas.
- Um procedimento de melhoria do gerenciamento de inventário, no qual um subconjunto das rotas é selecionado, em seguida é trocada a ordem de execução de duas rotas, uma que pertence a este subconjunto e a outra que não pertence. Com base no menor custo, esta troca é executada até que os tempos de trajeto de todas as rotas que pertencem ao subconjunto sejam trocadas.

Em (ARCHETTI; DOERNER; TRICOIRE, 2013) é apresentada uma heurística para resolver IRPTW que inclui restrições relacionadas ao agendamento de produção. Essa heurística é divida em duas fases. A primeira cria um plano da quantidade de entrega que tem como objetivo minimizar o período durante o qual pode ocorrer escassez. Este plano estabelece as quantidades a serem entregues a cada cliente em cada período. Inicialmente, é aplicada uma política para que seja entregue o menor número necessário de produtos em cada cliente. Desta forma, a próxima entrega é realizada no momento em que o cliente está sem estoque, com isso é reduzido o custo de armazenamento. A partir desta política, três heurísticas de construção são desenvolvidas para criar o plano de entrega:

- A primeira, gera uma entrega para cada cliente considerando suas demandas do período. Em seguida, para o excedente de produtos produzidos que não foram entregues, um subconjunto de clientes é identificado para receber tais produtos. Esse grupo de clientes é selecionado de maneira a reduzir o custo de entrega.
- A segunda heurística, executa todos os passos da primeira e, posteriormente, combina duas entregas no mesmo cliente sempre que possível.
- A terceira heurística ignora o passo de seleção do subconjunto da primeira heurística e utiliza a combinação de entregas da segunda. Isto é, todos os clientes irão receber os produtos excedentes e, se possível, entregas em um mesmo cliente serão combinadas.

Esse plano é, então, fixado e o problema de roteamento com janelas de tempo resultante pode ser resolvido na segunda fase com o algoritmo *Large Neighborhood Search* desenvolvido em (SHAW, 1998).

Em (JIA et al., 2014), é apresentado um modelo de programação inteira mista para resolver um IRP considerando custo de carregamento e janelas de tempo fracas durante o estágio de transporte. Outro objetivo considerado nesse estudo é determinar o plano de produção do fornecedor. Assim, como os outros autores, a abordagem heurística desenvolvida em (JIA et al., 2014) é dividida em duas etapas, a saber:

- A primeira etapa, utiliza busca Tabu para obter a matriz de entregas dos clientes. Essa matriz é considerada como o problema principal desse método, pois a partir dessas informações é possível organizar o agendamento de produção dos produtos. Para isso, a matriz é inicializada considerando que todos os clientes recebam apenas uma única entrega, considerando a demanda de cada período. Em seguida, os problemas

de agendamento de produção e roteamento são solucionados com seus respectivos algoritmos e o custo da solução inicial é obtido.

- A segunda fase tem como objetivo resolver dois subproblemas, o agendamento de produção e o roteamento de distribuição. Nesse caso, são utilizados um algoritmo de economias e um de busca de vizinhança com 2-opt, respectivamente.

O algoritmo prossegue de maneira cíclica, de modo que em cada iteração, a primeira fase é repetida e a busca tabu é realizada na matriz buscando suas soluções vizinhas. Uma solução vizinha utilizada na busca Tabu é um conjunto de outras soluções que podem ser alcançados a partir da solução atual, realizando um ou mais movimentos. Cada movimento da busca é realizado ao alterar um valor da matriz de entrega. Em cada iteração, a lista Tabu é atualizada com os movimentos proibidos para evitar que a solução pare em um ótimo local. A cada mudança na matriz de entregas, a fase dois é novamente executada e os subproblemas são solucionados considerando as mudanças na matriz de entregas. O custo da nova solução é obtido e utilizado para avaliar a busca Tabu. Por fim, a solução com menor custo identificada durante a busca é considerada a solução final.

Em (XIAO; RAO, 2016), um modelo matemático é apresentado para o IRP para multi-produtos em multi-períodos com restrições de janelas de tempo. Neste modelo, tanto os custos de inventário quanto os custos de distribuição são considerados. Um algoritmo genético com um módulo de lógica *fuzzy* é proposto para resolver o IRTWP. O cromossomo nesse modelo genético é uma sequência de dígitos binários que representam os clientes atendidos por cada centro de distribuição, i.e., o cromossomo modela uma solução do problema.

Em (IASSINOVSKAIA; LIMBOURG; RIANE, 2017) é proposta uma formulação de programação inteira mista linear para o IRPTW com itens de transporte retornáveis vazios e carregados. Para resolver problemas maiores é adicionada à solução uma heurística de agrupamento de clientes antes de executar o algoritmo *branch-and-cut* na formulação de programação inteira mista linear. O algoritmo heurístico de agrupamento atribui previamente um conjunto de clientes a cada veículo considerando algumas características comuns como por exemplo padrões de demanda e locação. Após fazer esse agrupamento, o problema IRPTW é resolvido para cada veículo e dessa forma o tempo do algoritmo é reduzido.

Diferente dos outros estudos, Li et al. (2014) têm apenas o objetivo de minimizar o tempo de trajeto das rotas. Uma vez que esse trabalho também apresenta um algoritmo de relaxação lagrangeana, ele será explicado na próxima seção.

4.5.2 Algoritmo de busca Tabu e algoritmo baseado em relaxação lagrangeana para IRPTW

O modelo de uma solução para o problema IRPTW proposta por Li et al. (2014) tem seus parâmetros, variáveis, restrições e função objetivo descritos a seguir. Os parâmetros utilizados são:

- M: valor muito grande para eliminar subrotas;
- b_j: horário de inicio da janela de atendimento do cliente $j \in S$;
- e_j: horário de fim da janela de atendimento do cliente $j \in S$;
- C_j: capacidade de estoque do cliente $j \in S$;
- I_j: inventário inicial do cliente $j \in S$;
- u_j: taxa de consumo do produto do cliente $j \in S$;
- R: taxa de tempo de entrega por unidade de produto;

As variáveis de decisão consideradas nesse modelo são:

- t_{jv}: horário de chegada do veículo $v \in V$ e início de atendimento do cliente $j \in S$;
- t_{0v}: horário de chegada do veículo $v \in V$ no depósito $0 \in S$;
- t^d_{jv}: horário de saída do veículo $v \in V$ do cliente $j \in S$;
- x_{ijv}: variável binária com valor 1 quando $j \in S$ é visitado imediatamente depois de $i \in S$ pelo veículo $v \in V$, e 0 caso contrário;
- q_{jv}: quantidade do produto entregue para o cliente $j \in \zeta$ no veículo $v \in V$;

A definição formal do modelo com sua função objetivo e restrições é:

$$min(\max_{v}\ t_{0v}) \tag{4}$$

$s.a.:$

$$\sum_{v\in V}\sum_{j\in\zeta, j\neq i} x_{ijv} = 1, \quad \forall i \in \zeta, \tag{5}$$

$$\sum_{j\in\zeta} x_{0jv} = 1, \quad \forall v \in V, \tag{6}$$

$$\sum_{j\in\zeta} x_{j0v} = 1, \quad \forall v \in V, \tag{7}$$

$$\sum_{i\in\zeta} x_{ihv} - \sum_{i\in\zeta} x_{hjv} = 0, \quad \forall h \in \zeta, v \in V, \tag{8}$$

$$t_{jv} \geq t^d_{iv} + \theta_{ij} - M(1 - x_{ijv}), \quad \forall i \in \zeta, j \in \zeta, v \in V, \tag{9}$$

$$\sum_{i\in\zeta} x_{ijv} b_j \leq t_{jv} \leq \sum_{i\in\zeta} x_{ijv} e_j, \quad \forall j \in \zeta, v \in V, \tag{10}$$

$$q_{iv} = \sum_{j\in\zeta} x_{ijv}(C_i - I_i + u_i t_{iv}), \quad \forall i \in \zeta, v \in V, \tag{11}$$

$$t^d_{0v} = 0, \quad \forall v \in V, \tag{12}$$

$$t^d_{iv} = t_{iv} + q_{iv}/R, \quad \forall i \in \zeta, v \in V, \tag{13}$$

$$t_{iv} \geq 0, t^d_{iv} \geq 0, \quad \forall i \in \zeta, v \in V, \tag{14}$$

Como já comentado, diferente dos outros métodos a função objetivo (4) busca apenas minimizar o tempo total máximo de trajeto dos veículos. A Restrição (5) garante que cada cada cliente só seja visitado uma única vez. As Restrições de (6)–(8) constroem a sequencia da rota de cada veículo. A Restrição (9) tem como objetivo eliminar subrotas derivadas do tempo de trajeto de cada, para isso é considerado um valor muito grande M. A Restrição (10) especifica que o horário no qual o veículo irá chegar no cliente j estará dentro da janela de atendimento definida. A quantidade de abastecimento em cada cliente é determinado pela Restrição (11), a Restrição (12) define o horário em que o veículo irá sair do centro de distribuição e a Restrição (13) define os horários que o veículo irá sair de cada cliente. Por fim a Restrição (14) define a integridade do horário de início e fim dos atendimentos de cada período.

Neste trabalho novamente o algoritmo de busca Tabu é adaptado para o problema IRPTW. Outra contribuição deste trabalho é propor um modelo de otimização robusta para encontrar um limite inferior utilizando o método de relaxação Lagrangeana. Os

autores demonstram que a busca Tabu é capaz de fornecer uma solução quase ótima, aproximando-se do limite inferior encontrado.

Algoritmo de busca Tabu

No algoritmo de busca Tabu proposto, uma solução inicial é criada sequenciando os clientes em ordem crescente do fim da janela de atendimento. Em seguida, os clientes são inseridos em rotas de veículos de maneira gulosa até que todos os clientes estejam em uma rota. Após a construção dessa solução inicial, o algoritmo US (GENDREAU; HERTZ; LAPORTE, 1992) é adaptado para melhorar a solução inicial e também as soluções vizinhas criadas durante a busca Tabu. O método continua a busca para melhorar a qualidade das soluções vizinhas em cada iteração da heurística. Estas soluções vizinhas são geradas trocando partes de duas rotas na solução origem.

O algoritmo US foi proposto em (GENDREAU; HERTZ; LAPORTE, 1992) como uma melhoria da solução de um TSP. O algoritmo US foi adaptado para aprimorar a solução encontrada pela busca Tabu considerando as restrições do IRPTW. Nesta adaptação, para cada rota há (i) restrições de janela de tempo para clientes e (ii) restrição de hora de operação máxima para cada veículo. Assim, pode haver menos clientes em cada rota, em comparação com a VRP padrão para problemas de tamanho semelhante. Os autores adaptaram a seleção de clientes do algoritmo original para considerar apenas os dois clientes mais próximos do ponto de inserção.

Relaxação Lagrangeana

Com o objetivo de avaliar o desempenho do algoritmo de Busca Tabu para IRPTW, os autores visam encontrar os limites inferiores do modelo proposto, utilizando a técnica de relaxação lagrangeana (explicada na Seção 2). Como restrição complexa, é selecionada a restrição (5) que garante que cada cliente só é visitado uma única vezes por período. Os multiplicadores lagrangeanos μ são criados resultando no modelo a seguir:

$$min((\max_{k} t_{0k}) + \sum_{i \in \zeta} \mu_i (\sum_{v \in V} \sum_{j \in \zeta} x_{ijv} - 1)) \tag{15}$$

$$s.a. : (6) - (14)$$

Após a relaxação, os autores conseguem decompor o problema em um conjunto de subproblemas, um para cada veículo, cada um dos quais pode ser tratado como um problema de TSP com restrições de tempo. Por fim, o algoritmo de subgradiente é usado para encontrar os multiplicadores lagrangeanos para o modelo proposto.

5 Algoritmo de Rahim para o SIRP

Em (RAHIM et al., 2014), é proposta a utilização da relaxação lagrangeana para determinar uma solução viável para o problema SIRP considerando múltiplos períodos. A Figura 1 apresenta todos os passos da modelagem e solução proposta em (RAHIM et al., 2014). Primeiro, é feita a modelagem estocástica, em que o SIRP é modelado como um problema de programação inteira mista e estocástica (PIME). Depois, é criado um novo modelo do problema, traduzindo as variáveis estocásticas para uma aproximação determinística. Posteriormente, o problema é decomposto em dois subproblemas, o subproblema de estoque e o subproblema de roteamento de veículos usando os multiplicadores lagrangeanos (passo descrito na Seção 5.3). E, finalmente, um limite inferior e superior para o problema determinístico é encontrado usando o algoritmo subgradiente para atualizar os multiplicadores lagrangeanos. Esse último passo de cor rosa na Figura 1 é descrito na Seção 5.4.

5.1 Programação inteira mista e estocástica

As variáveis, parâmetros, restrições e função objetivo da PIME para modelar o SIRP, proposto por (RAHIM et al., 2014), são descritos a seguir. Seja $H = \{1, 2, \cdots, T\}$ o conjunto de períodos consecutivos e $H^+ = H \cup \{0\}$, o parâmetro τ_h representa o número de horas em um período h em que $h \in H^+$. Seja ζ o conjunto de clientes e $S = \zeta \cup \{r\}$, em que $r = 0$ é o depósito e os clientes são numerados de 1 até N. A frota de veículos é homogênea e cada veículo $v \in V$ tem capacidade k_v. Para simplificar o entendimento do modelo proposto, vamos considerar que o conjunto S^2 é a combinação de todos os pares (i, j), em que $i, j \in S$. Os outros parâmetros do modelo são:

- R_{rh}: quantidade de produto a ser reabastecida no estoque do depósito no período $h \in H$;
- d_{jh}: demanda estocástica do cliente $j \in \zeta$ por hora do período $h \in H$. Os autores assumem que essa demanda tem uma distribuição normal em que $D_j = E(d_{jh})$ e o desvio padrão é igual a σ_j;
- η_{jh} : custo de armazenamento de cada unidade do produto para $j \in S$ no período $h \in H$;

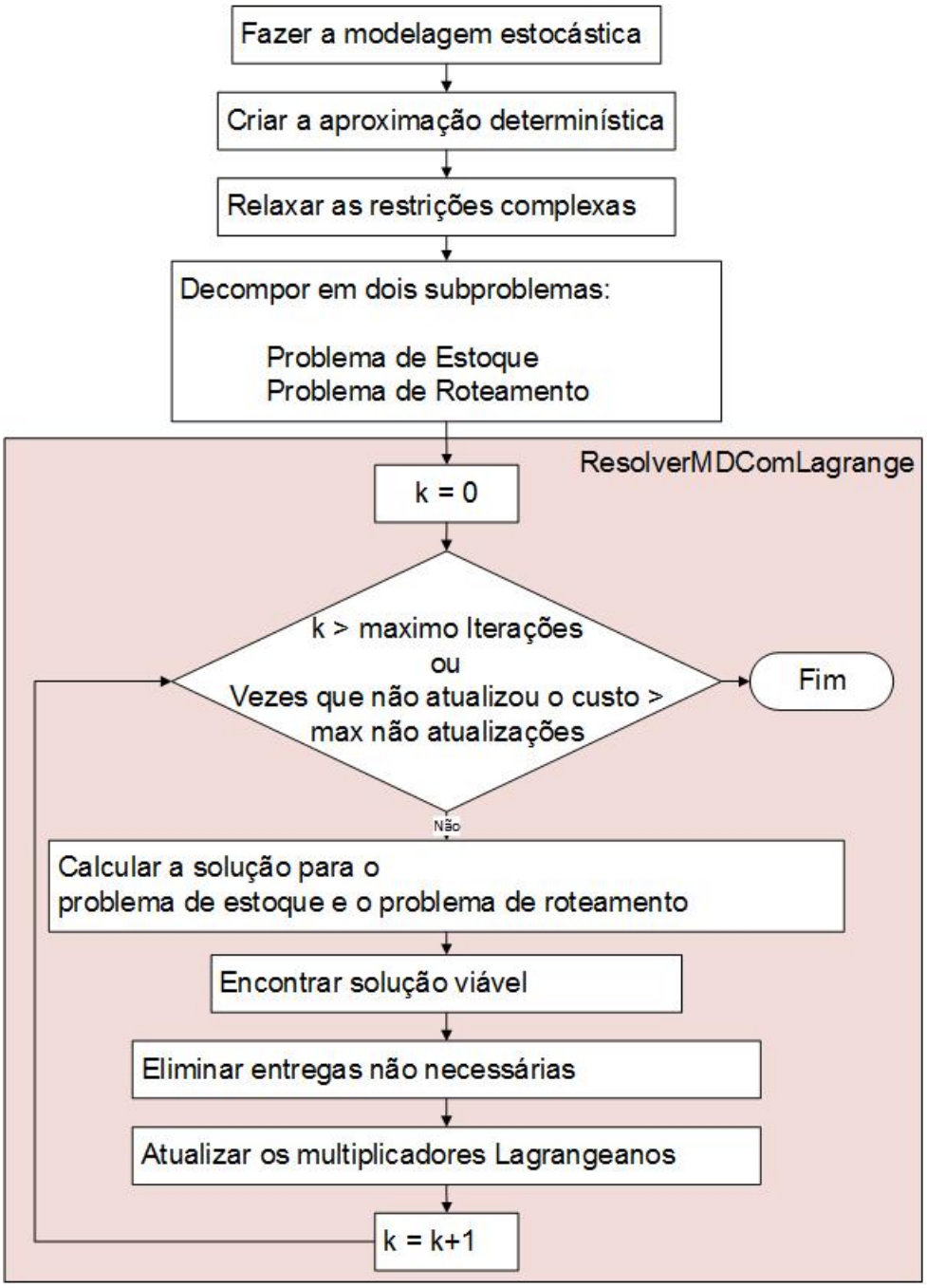

Figura 1 – Passos da solução proposta em (RAHIM et al., 2014).

- ψ_v: custo operacional do veículo $v \in V$;
- φ_{jh}: custo fixo de entrega para $j \in S$ no período $h \in H$;
- δ_v: custo de viagem do veículo $v \in V$ por km;
- ν_v: velocidade média do veículo $v \in V$ (em km por hora);
- θ_{ij}: duração da viagem entre o par $(i, j) \in S^2$ (em horas).

Assim, a demanda estocástica do cliente $j \in \zeta$ no período $h \in H$ é $\Lambda_{jh} = d_{jh}\tau_h$. Considerando os últimos quatro parâmetros, o custo de transporte e de entrega de

uma viagem entre o par $(i,j) \in S^2$ usando o veículo $v \in V$ no período $h \in H$ é $\Gamma_{ijhv} = \delta_v \nu_v \theta_{ij} + \varphi_{jh}$. As variáveis consideradas no modelo proposto por Rahim et al. (2014) são:

- x_{ijhv}: variável binária com valor 1 quando $j \in S$ é visitado imediatamente depois de $i \in S$ pelo veículo $v \in V$ no período $h \in H$, e 0 caso contrário;
- y_{hv}: variável binária com valor 1 quando o veículo $v \in V$ é utilizado no período $h \in H$, e 0 caso contrário;
- q_{jh}: quantidade do produto entregue para o cliente $j \in \zeta$ no período $h \in H$;
- I_{jh}: nível de estoque de $j \in S$ no final do período $h \in H^+$;
- Q_{ijhv}: quantidade do produto restante no veículo $v \in V$ quando visita $j \in S$ imediatamente após $i \in S$ no período $h \in H$. Quando não existe a viagem (i,j), o valor dessa variável é zero.

O modelo de PIME que resolve o SIRP é:

$$min \sum_{h \in H} \sum_{v \in V} \left(\psi_v y_{hv} + \sum_{(i,j) \in S^2} \Gamma_{ijhv} x_{ijhv} \right) + \sum_{h \in H^+} \sum_{j \in S} \eta_{jh} I_{jh} \tag{16}$$

$s.a.:$

$$\sum_{v \in V} \sum_{i \in S} x_{ijhv} \leq 1, \quad \forall j \in \zeta, h \in H, \tag{17}$$

$$\sum_{i \in S} (x_{ijhv} - x_{jihv}) = 0, \quad \forall j \in S, h \in H, v \in V, \tag{18}$$

$$\sum_{(i,j) \in S^2} \theta_{ij} x_{ijhv} \leq \tau_h, \quad \forall h \in H, v \in V, \tag{19}$$

$$\sum_{v \in V} \sum_{i \in S} (Q_{ijhv} - Q_{jihv}) = q_{jh}, \quad \forall j \in \zeta, h \in H, \tag{20}$$

$$Q_{jkhv} \leq k_v x_{jkhv}, \quad \forall i,j \in S, h \in H, v \in V, \tag{21}$$

$$I_{r,h-1} + R_{rh} - I_{r,h} = \sum_{j \in \zeta} q_{jh}, \quad \forall j \in \zeta, h \in H, \tag{22}$$

$$I_{j,h-1} + q_{jh} - I_{j,h} = \Lambda_{jh}, \quad \forall j \in \zeta, h \in H, \tag{23}$$

$$I_{j0} \leq I_{jT}, \quad \forall j \in \zeta, \tag{24}$$

$$x_{rjhv} \leq y_{hv} \quad \forall j \in \zeta, h \in H, v \in V, \tag{25}$$

$$x, y \in \{0,1\}^{|S|^2|H||V|}, I \geq 0, Q \geq 0, q \geq 0, \tag{26}$$

em que a função objetivo é composta de três custos: (i) o custo de operação do veículo; (ii) o custo total de transporte e de entrega; e (iii) o custo de armazenamento do produto no final de cada período.

A Restrição (17) garante que no período h cada cliente seja visitado no máximo uma vez. A Restrição (18) assegura que a rota de um veículo deve começar no depósito, passar pelos clientes e retornar para o depósito, assegurando o ciclo da rota. A Restrição (19) garante que todo o tempo de trajeto da rota do veículo não ultrapasse as horas de trabalho do período. A Restrição (20) determina a quantidade de entrega para um cliente. A Restrição (21) assegura que a capacidade do veículo não vai ser ultrapassada. A Restrição (22) assegura o balanceamento do estoque do depósito e a Restrição (23) atende as demandas estocásticas dos clientes. Uma vez que não há custo para o inventário inicial, a Restrição (24) garante que o estoque de um determinado cliente no final do horizonte de planejamento não seja menor do que o do início do planejamento. E por fim, a Restrição (25) assegura que um veículo não pode atender um cliente, em um determinado período h, se o mesmo não for selecionado no período h.

Com relação às restrições, é importante notar que:

- No artigo original, a Restrição (22), multiplicava o valor de q_{jt} por τ_t. Porém, o valor da variável q_{jt} não é dada em horas.
- Com relação ao inventário inicial, existem duas possibilidades. A primeira considerada que o inventário inicial é um parâmetro e deve-se ter inventário inicial suficiente para garantir a viabilidade, essa opção é problemática pois o valor escolhido como inventário inicial pode influenciar na qualidade da solução e o tempo computacional. A segunda possibilidade é considerar o inventário inicial como uma variável, mas, como não há custo para o inventário inicial, é necessário que o inventário final seja tão alto quanto o inicial. Essa segunda opção é a escolhida pelos autores. Assim, a restrição (24) é necessária pois a dificuldade neste problema é ter uma solução viável no início do período.

5.2 Aproximação determinística

No modelo determinístico proposto por (RAHIM et al., 2014) que aproxima a PIME (Modelo (16)), a Restrição (23) é substituída por uma nova restrição que deve satisfazer a

demanda estocástica em cada período com um determinado nível de confiança $(1-\alpha)$. Para isso foi utilizado a desigualdade de Chebyshev (KVANLI; PAVUR; KEELING, 2005), garantindo que:

$$Probabilidade\left\{I_{j,h-1}+\sum_{s=h}^{T}q_{jh}\geq\sum_{s=h}^{T}d_{js}\tau_s\right\}=(1-\alpha),\quad\forall j\in S,h\in H$$

.

Essa nova restrição é:

$$I_{j,h-1}+\sum_{s=h}^{T}q_{js}\geq\sum_{s=h}^{T}E\left(d_{js}\right)\tau_s+z_\alpha\left(\sqrt{T-h+1}\right)\sigma_j,\forall j\in\zeta,h\in H. \tag{27}$$

A Restrição (27) previne a escassez de estoque em cada cliente com um nível de confiança $(1-\alpha)$. Assim, $100*(1-\alpha)\%$ do nível de serviço é garantido, definido pelo valor normal padrão z_α. O modelo determinístico resultante (MD) é:

$$min\sum_{h\in H}\sum_{\upsilon\in V}\left(\psi_\upsilon y_{h\upsilon}+\sum_{(i,j)\in S^2}\Gamma_{ijh\upsilon}x_{ijh\upsilon}\right)+\sum_{h\in H^+}\sum_{j\in S}\eta_{jh}I_{jh} \tag{28}$$

$$s.a:(17)-(22),(24),(25)-(27)$$

5.3 Usando relaxação lagrangeana

Em (RAHIM et al., 2014) é proposto um algoritmo que usa relaxação lagrangeana para decompor o problema e encontrar limitantes inferiores (LI) e superiores (LS) para o problema. A relaxação Lagrangeana consiste em relaxar algumas restrições de um modelo computacionalmente difícil de resolver e dessa forma o modelo computacionalmente complexo pode ser convertido para um modelo tratável. Essas restrições são eliminadas e incorporadas na função objetivo como uma penalidade. A restrição (20) foi escolhida por (RAHIM et al., 2014) para ser relaxada pois combina variáveis de estoque (q_{jh}) e variáveis de transporte $(Q_{ijh\upsilon})$. Assim, a restrição (20) é inserida na função objetivo com multiplicadores lagrangeanos μ_{jh} para todo $j\in S,h\in H$. Essa reformulação que usa relaxação lagrangeana, chamada de RL, é:

$$min \sum_{h\in H}\sum_{v\in V}\left(\psi_v y_{hv} + \sum_{(i,j)\in S^2}\Gamma_{ijhv}x_{ijhv}\right) + \sum_{h\in H^+}\sum_{j\in S}\eta_{jh}I_{jh}$$
$$+ \sum_{h\in H}\sum_{j\in\zeta}\mu_{jh}\left(q_{jh} - \sum_{v\in V}\sum_{i\in S}\left(Q_{ijhv} - Q_{jihv}\right)\right) \quad (29)$$
$$s.a : (17) - (19), (21), (22), (24), (25) - (27).$$

No Modelo (29) podemos formar dois conjuntos disjuntos de restrições considerando o tipo de variáveis usadas. As Restrições (22), (24) e (27) contém apenas variáveis de inventário e as restrições (17)-(19),(21) e (25) contém apenas variáveis de roteamento. Se observarmos a função objetivo, é possível também separar os termos em dois conjuntos, um que contém os termos das variáveis de inventário e outro com as variáveis de roteamento. Assim, o problema de IRP pode ser decomposto em um subproblema de estoque (I) e um subproblema de roteamento (R). Apesar de que o problema de roteamento e o problema de inventário resultantes são NP-difíceis (BITRAN; YANASSE, 1982), eles podem ser resolvidos na prática mais facilmente por um otimizador de programação inteira mista. Esses dois submodelos são apresentados a seguir.

O modelo do problema de inventário (RLI) é:

$$Z_{RLI} = min : \sum_{h\in H^+}\sum_{j\in S}\eta_{jh}I_{jh} + \sum_{h\in H}\sum_{j\in\zeta}\mu_{jh}q_{jh} \quad (30)$$
$$s.a : (22), (24)\, e\, (27).$$

O modelo do problema de roteamento (RLR) é:

$$Z_{RLR} = min : \sum_{h\in H}\sum_{v\in V}\left(\psi_v y_{hv} + \sum_{(i,j)\in S^2}\Gamma_{ijhv}x_{ijhv}\right) -$$
$$\sum_{h\in H}\sum_{j\in\zeta}\mu_{jh}\left[\sum_{v\in V}\sum_{i\in S}\left(Q_{ijhv} - Q_{jihv}\right)\right] \quad (31)$$
$$s.a : (17) - (19), (21)\, e\, (25)$$
$$E(d_{j1})\tau_1 + z_\alpha\sigma_j - I_{j0} \leq \sum_{v\in V}\sum_{i\in S}Q_{ij1v}, \quad \forall j \in \zeta. \quad (32)$$

A Restrição (32) garante com certo nível de confiança que não ocorrerá escassez em cada cliente durante o primeiro período no horizonte de planejamento.

Partindo da decomposição descrita anteriormente, um limitante inferior (LI) para o Modelo (28) pode ser encontrado para qualquer vetor de multiplicadores lagrangeano μ. O melhor LI pode ser calculado a partir do vetor de multiplicadores lagrangeano ótimo que é a solução do dual do Modelo (29), i.e.:

$$max \quad L(\mu_{jt}), \tag{33}$$

em que $L(\mu_{jt})$ é:

$$\begin{aligned} min \sum_{h\in H}\sum_{v\in V}\left(\psi_v y_{hv} + \sum_{(i,j)\in S^2}\Gamma_{ijhv}x_{ijhv}\right) + \sum_{h\in H^+}\sum_{j\in S}\eta_{jh}I_{jh} \\ + \sum_{h\in H}\sum_{j\in\zeta}\mu_{jh}\left[q_{jh} - \sum_{v\in V}\sum_{i\in S}(Q_{ijhv} - Q_{jihv})\right] \end{aligned} \tag{34}$$

5.4 Resolvendo o problema dual

Para resolver o Modelo (33), primeiramente os subproblemas de inventário (Modelo (30)) e de roteamento (Modelo (31)) são solucionados para encontrar um limitante inferior, e então o algoritmo de subgradiente é utilizado para ajustar o valor de μ_{jt} ao longo do tempo para encontrar valores que produzem melhores limitantes inferiores. Além disso, uma heurística que recebe as soluções da relaxação lagrangeana é usada para compor uma solução viável do problema original.

5.4.1 Algoritmo subgradiente

O processo de otimização do subgradiente (Algoritmo 1) que ajusta μ_{jt} gera limitantes inferiores e superiores a cada iteração do algoritmo, atualizando o melhor limite inferior e superior até o momento. Dado um valor inicial μ^0, uma sequência de multiplicadores lagrangeanos é gerada usando a equação:

$$\mu^{k+1} = \mu^k + s^k g^k, \tag{35}$$

em que g^k é a matriz subgradiente e s^k é o tamanho do passo para atualização de μ. Essa matriz é calculada usando a seguinte equação:

$$g^k_{jh} = q_{jh} - \sum_{v\in V}\sum_{i\in S}\left(Q_{ijhv} - Q_{jihv}\right), \tag{36}$$

em que $j \in \zeta$ e $h \in H$; e q_{jh}, Q_{ijhv} e Q_{jihv} são os valores na solução encontrada para o problema na iteração k. O tamanho do passo é determinado considerando a seguinte equação:

$$s^k \leftarrow \omega(LS - Ck_{LI})/(\parallel g^k \parallel)^2, \tag{37}$$

em que ω é a *agilidade* do subgradiente, LS é um limitante superior para a solução ótima do problema original que pode ser encontrada aplicando uma heurística e Ck_{LI} é o custo usando os multiplicadores lagrangeanos na iteração k. Se $s^k \to 0$ e $\sum_{i=0}^{k} s^k \to \infty$ então Ck_{LI} converge para a solução do Modelo (33). Garantir essas condições pode ser uma tarefa difícil. Uma opção é inicializar ω com 2 e dividir esse valor por 2 quando o valor CK_{LI} não for melhorado. Essa opção apresenta bons resultados na prática, porém não garante as condições para a convergência (FISHER, 1981).

O Algoritmo 1 recebe como parâmetros o número máximo de iterações, o número máximo de não atualizações do custo que pode ocorrer e os parâmetros do problema $\mathcal{P} = \langle H^+, \tau, S, V, k, R, d, \eta, \psi, \varphi, \delta, \nu, \theta, \Lambda, \Gamma \rangle$. A saída do algoritmo é a melhor solução encontrada $\mathcal{R} = \langle x, y, I, Q, q \rangle$. O algoritmo começa inicializando as variáveis LI, LS, ω e μ, essas duas últimas variáveis com valores iniciais entre zero e um (Linhas 2–5). Após a inicialização, $\mathcal{R}^*$ (a melhor solução viável encontrada até o momento) é calculada resolvendo o modelo MD (Modelo (28)) para um cenário em que cada cliente é atendido separadamente por um veículo (Linhas 7–8). O primeiro valor para LS é o custo obtido usando essa solução (Linha 9).

Em seguida, o algoritmo calcula LI, LS e atualiza os multiplicadores lagrangeanos μ em cada iteração (Linhas 10–32). O algoritmo só irá ser finalizado quando o número máximo de iterações for alcançado (Linha 10) ou o custo Ck_{LI} não for melhorado certo número de vezes (Linhas 31–32).

Para ajustar o LI são resolvidos o subproblema de inventário (Modelo 30) e o subproblema de roteamento (Modelo 31) (Linhas 12 e 13). Se a somatória das soluções para esses dois subproblemas for maior que o LI atual, então LI é atualizado com esse novo valor, caso contrário ω é atualizado (Linhas 15–18).

Para ajustar o LS, uma heurística é aplicada para encontrar uma solução viável baseando-se no resultado obtido na solução dos dois subproblemas (Linha 20). Caso essa solução melhore o LS atual, então a solução encontrada é considerada a melhor solução até o momento (Linhas 22-24). Após isso, um procedimento de ajuste é aplicado para eliminar

entregas não necessárias e LS é atualizado (Linhas 25–26). E por fim, os multiplicadores lagrangeanos μ são atualizados (Linhas 28–30).

Algoritmo 1 ResolverMDComLagrange (RAHIM et al., 2014)

Require: $maxIteracoes, maxNaoAtualizacoes, instancia$

1: *//************************Inicializando Variáveis*
2: $LI \leftarrow 0$
3: $LS \leftarrow \infty$
4: $\omega \leftarrow Random(0, 1)$
5: $\mu^k \leftarrow$ obterMatrizAleatória$(0, 1)$
6: *//********************************Inicializando LS*
7: $PSimples \leftarrow$ criarCenarioUmClientePorVeiculo$(\mathcal{P})$
8: $\mathcal{R}^* \leftarrow$ solução do modelo MD para PSimples
9: $LS \leftarrow$ custo da solução $\mathcal{R}^*$ de MD
10: **for** $k = 1$ to $maxI$ **do**
11: *//********************************Atualizando LI*
12: $\mathcal{R}_R \leftarrow$ solução do modelo RLR com μ^k para $\mathcal{P}$
13: $\mathcal{R}_I \leftarrow$ solução do modelo RLI com μ^k para $\mathcal{P}$
14: $Ck_{LI} \leftarrow$ custo da solução $\mathcal{R}_R$ de RLR + custo da solução $\mathcal{R}_I$ de RLI
15: **if** $Ck_{LI} > LI$ **then**
16: $LI \leftarrow Ck_{LI}$
17: **else**
18: $\omega \leftarrow \omega/2$
19: *//********************************Atualizando LS*
20: $\mathcal{R}^k \leftarrow$ encontrarSoluçaoViável$(\mathcal{R}_I, \mathcal{P})$
21: $Ck_{LS} \leftarrow$ custo da solução $\mathcal{R}^k$ de MD
22: **if** $Ck_{LS} > LS$ **then**
23: $LS \leftarrow Ck_{LS}$
24: $\mathcal{R}^* \leftarrow \mathcal{R}^k$
25: $\mathcal{R}^* \leftarrow$ eliminarEntregasNãoNecessárias$(\mathcal{R}^*, \mathcal{P})$
26: $LS \leftarrow$ custo da solução $\mathcal{R}^*$ de MD
27: *//********************************Atualizando μ*
28: $g^k \leftarrow$ calcularSubgradiente$(\mathcal{R}^*)$
29: $s^k \leftarrow \omega(LS - Ck_{LI})/(\| g^k \|)^2$
30: $\mu^k \leftarrow \mu^k + s^k g^k$
31: **if** vezes não atualizou $Ck_{LI} >$ maxNA **then**
32: **break**
return $\mathcal{R}^*$

Fonte: (RAHIM et al., 2014)

5.4.2 Encontrando uma solução viável

O algoritmo 2 recebe a solução encontrada do modelo RLI e os parâmetros $\mathcal{P}$. A informação de quantidade que deve ser entregue em cada cliente durante cada período é

utilizada para resolver o problema de roteamento. Esse problema é resolvido para cada período separadamente, utilizando o método heurístico de economias conhecido como Clarke & Wright (descrito na seção 3.1.1). O algoritmo também verifica que a solução encontrada não viole as restrições (19) e (21).

O conjunto SR_h (Linha 3) representa o conjunto de todos os clientes que têm uma entrega a ser realizada no período h e a variável Rota (Linha 4) é uma lista temporária que irá conter inicialmente os tours básicos para cada cliente. As variáveis SR_h e Rota são construídas respeitando a restrição do tempo de trajeto total do período τ_h (Linhas 5–10). O procedimento para encontrar uma solução viável baseado em (CLARKE; WRIGHT, 1964) é executado enquanto for possível encontrar uma economia na solução (Linhas 12–39). Nesse procedimento, dois tours (i, j) já existentes na lista *Rota* são combinados em um único tour e a melhor combinação viável é armazenada, i.e., aquela combinação com melhor economia (Linhas 18–34). Para cada combinação, um algoritmo para o problema de TSP é utilizado para encontrar a melhor sequência de entrega para esse novo tour (Linhas 20–21). Caso o tempo desse novo trajeto viole a restrição do tempo de trabalho do período h e as outras restrições do problema, a combinação dos tours (i, j) é desconsiderada (Linhas 23–24), caso contrário, a economia desse novo tour é calculada e se for maior do que a melhor economia encontrada até o momento, esse tour é armazenado e a melhor economia é atualizada (Linhas 26–34). Após testar todas as combinações de tours, se houve uma melhoria na economia, então são removidos os tours i e j originais e adicionado o novo tour na lista *Rota* (Linhas 35–38).

Quando não houver mais melhoria nessa solução, é calculado o custo total da lista *Rota* e o número de veículos necessários para atender essa *Rota*, essa rota é adicionada como solução do período h (Linha 40–42).

5.4.3 Eliminando entregas não necessárias

O Algoritmo 3 tenta combinar duas ou mais entregas em todo o horizonte de planejamento, de um determinado cliente, em uma única entrega. Se essa combinação resulta em uma solução melhor do que a atual, isto é com, menor custo, então ela é considerada a melhor solução até o momento.

Algoritmo 2 EncontrarSoluçaoViável

Require: $(X, Z)_I, (X, Z)_R, instancia$

1: $(X_k, Z_k) \leftarrow nil$
2: $(X^*, Z^*) \leftarrow (X, Z)_I$
3: **for all** $h \in H$ **do**
4: $SR_h \leftarrow \{\}$ ▷ Inicializando a rota
5: $Rota \leftarrow nil$
6: **for all** $j \in S$ **do**
7: $tempoTrajetodeposito \leftarrow tempo(deposito, j) + tempo(j, deposito)$
8: **if** $(X^*, Z^*).q_{jh} > 0 \ \wedge tempoTrajetodeposito < \tau_h$ **then**
9: $SR_h \leftarrow SR_h \cup \{j\}$
10: $Tour \leftarrow \{j\}$
11: $Rota.add\,(Tour)$
12: $melhorEconomia \leftarrow -1$
13: **repeat** ▷ Combinando tours
14: $n \leftarrow tamanho(Rota)$
15: $melhorTouri \leftarrow -1$
16: $melhorTourj \leftarrow -1$
17: $melhorTour \leftarrow \{\}$
18: $TourC^+ \leftarrow \{\}$
19: **for** $i = 0$ to n **do**
20: **for** $j =$ i+1 to n **do**
21: $TourC^+ \leftarrow Rota[i] \cup Rota[j]$
22: TSP$(TourC^+)$
23: $tempoTour \leftarrow calculaTempoTrajeto(TourC^+)$
24: **if** $tempoTour > \tau_h \wedge naoValidaRestricoes(TourC^+)$ **then**
25: $Desconsidera(i, j)$
26: **else**
27: $tempoTrajetoi \leftarrow calculaTempoTrajeto(Rota[i])$
28: $tempoTrajetoj \leftarrow calculaTempoTrajeto(Rota[j])$
29: **if** $tempoTour < tempoTrajetoi + tempoTrajetoj$ **then**
30: $SV \leftarrow tempoTrajetoi + tempoTrajetoj - tempoTour$
31: **if** $SV > melhorEconomia$ **then**
32: $melhorEconomia \leftarrow SV$
33: $melhorTour \leftarrow TourC^+$
34: $melhorTouri \leftarrow i$
35: $melhorTourj \leftarrow j$
36: **if** $melhorTouri > -1 \wedge melhorTourj > -1$ **then**
37: $Rota.remove(melhorTouri)$
38: $Rota.remove(melhorTourj)$
39: $Rota.add(melhorTour)$
40: **until** $melhorEconomia > 0$
41: calculaCustoRota$(Rota)$
42: calculanúmeroVeículos$(Rota)$
43: $(X_k, Z_k).addRotaXZ(Rota)$
return (X_k, Z_k)

Fonte: (RAHIM et al., 2014)

Primeiro, é feita uma cópia da solução atual (Linha 1) e calculado o conjunto W que contém os clientes que têm mais de uma entrega em todo o horizonte (Linhas 2–9). Em seguida, o algoritmo irá tentar combinar duas entregas em uma única entrega. Então, para cada cliente j no conjunto W (Linha 10), o procedimento irá começar do período final do horizonte de planejamento até o período inicial (Linha 12).

Em cada período, se a soma da quantidade de entrega dele com o período seguinte não ultrapassar a capacidade do veículo (Linha 17), então as duas entregas são agrupadas no mesmo período e atualizados os estoques do cliente j (Linhas 18–21). Para esse novo escalonamento de entregas, o algoritmo EncontrarSoluçaoViável é aplicado (Linha 22), caso a solução gerada tenha um custo melhor do que a atual, a solução gerada se torna a atual (Linhas 23–24).

Em seguida, o algoritmo irá verificar novamente se há algum cliente que tenha mais de uma entrega em todo o horizonte de planejamento (Linhas 25–28). Caso ainda haja algum cliente com mais de uma entrega, é realizado um procedimento parecido com o anterior, porém agora começando no início do horizonte e iterando até o final do planejamento, tentando agrupar duas entregas em um único período (Linhas 29–42). Caso esse procedimento melhore o custo da solução, ela é considerada como a melhor solução até o momento (Linhas 43-44).

Neste trabalho são propostas duas extensões do trabalho de Rahim et al. (2014). A primeira, para melhorar o tempo computacional e a qualidade das soluções. Para isso será proposta a adaptação da heurísticas para encontrar soluções factíveis. A segunda extensão será a inclusão e adaptação de restrições no modelo, com o objetivo de aproximar o modelo de um cenário mais realista.

Algoritmo 3 EliminarEntregasNãoNecessárias

Require: $(X^*, Z^*), instancia$

1: $(X_k, Z_k) \leftarrow (X^*, Z^*)$
2: $W \leftarrow \{\}$
3: **for all** $i \in S$ **do**
4: $\quad C \leftarrow 0$
5: $\quad$ **for all** $h \in H$ **do**
6: $\quad\quad$ **if** $(X^*, Z^*).q_{jh} > 0$ **then**
7: $\quad\quad\quad C \leftarrow C + 1$
8: $\quad$ **if** $C > 1$ **then**
9: $\quad\quad W \leftarrow W \cup \{i\}$
10: **for all** $j \in W$ **do**
11: $\quad W \leftarrow W \setminus \{j\}$
12: $\quad$ **for** $m =$ h-1 to *0* **do**
13: $\quad\quad$ **if** $(X^*, Z^*).q_{jm} > 0$ **then**
14: $\quad\quad\quad$ **if** $(X^*, Z^*).q_{jm} < k_v$ **then**
15: $\quad\quad\quad\quad (X_k, Z_k).q_{jm} \leftarrow (X^*, Z^*).q_{jm}$
16: $\quad\quad\quad\quad$ **for** $n =$ m+1 to h **do**
17: $\quad\quad\quad\quad\quad$ **if** $(X^*, Z^*).q_{jn} + (X_k, Z_k).q_{jm} < k_v$ **then**
18: $\quad\quad\quad\quad\quad\quad (X_k, Z_k).q_{jm} \leftarrow (X_k, Z_k).q_{jm} + (X^*, Z^*).q_{jn}$
19: $\quad\quad\quad\quad\quad\quad (X_k, Z_k).q_{jn} \leftarrow 0$
20: $\quad\quad\quad\quad\quad\quad (X_k, Z_k).I_{jm} \leftarrow (X_k, Z_k).I_{jm} + (X^*, Z^*).q_{jn}$
21: $\quad\quad\quad\quad\quad\quad (X_k, Z_k).I_{jn} \leftarrow (X_k, Z_k).I_{jn} - (X^*, Z^*).q_{jn}$
22: $\quad\quad\quad (X_k, Z_k) \leftarrow$ EncontrarSoluçaoViável$((X_k, Z_k), instancia)$
23: $\quad\quad\quad$ **if** $custo(DP, (X_k, Z_k)) < custo(DP, (X^*, Z^*))$ **then**
24: $\quad\quad\quad\quad (X^*, Z^*) \leftarrow (X_k, Z_k)$
25: $\quad C \leftarrow 0$
26: $\quad$ **for all** $h \in H$ **do**
27: $\quad\quad$ **if** $(X^*, Z^*).q_{jh} > 0$ **then**
28: $\quad\quad\quad C \leftarrow C + 1$
29: $\quad$ **if** $C > 1$ **then**
30: $\quad\quad$ **for** $n = 2$ to h **do**
31: $\quad\quad\quad$ **if** $(X^*, Z^*).q_{jn} > 0$ **then**
32: $\quad\quad\quad\quad$ **if** $(X^*, Z^*).q_{jn} < k_v$ **then**
33: $\quad\quad\quad\quad\quad (X_k, Z_k).q_{jn} \leftarrow (X^*, Z^*).q_{jn}$
34: $\quad\quad\quad\quad\quad (X_k, Z_k).I_{j0} \leftarrow (X^*, Z^*).I_{j0}$
35: $\quad\quad\quad\quad\quad$ **for** $m = 1$ to n **do**
36: $\quad\quad\quad\quad\quad\quad$ **if** $(X^*, Z^*).q_{jm} > 0$ **then**
37: $\quad\quad\quad\quad\quad\quad\quad$ **if** $(X^*, Z^*).q_{jm} + (X_k, Z_k).q_{jn} < k_v$ **then**
38: $\quad\quad\quad\quad\quad\quad\quad\quad (X_k, Z_k).q_{jn} \leftarrow (X_k, Z_k).q_{jn} + (X^*, Z^*).q_{jm}$
39: $\quad\quad\quad\quad\quad\quad\quad\quad (X_k, Z_k).I_{j0} \leftarrow (X_k, Z_k).I_{j0} + (X^*, Z^*).q_{jm}$
40: $\quad\quad\quad\quad\quad\quad\quad\quad (X_k, Z_k).q_{jm} \leftarrow 0$
41: $\quad\quad\quad\quad\quad\quad\quad\quad (X_k, Z_k).I_{jn} \leftarrow (X_k, Z_k).I_{jn} + (X^*, Z^*).q_{jm}$
42: $\quad\quad\quad\quad\quad\quad\quad\quad (X_k, Z_k).I_{jm} \leftarrow (X_k, Z_k).I_{jm} - (X^*, Z^*).q_{jm}$
43: $\quad\quad\quad\quad$ **if** $custo(DP, (X_k, Z_k)) < custo(DP, (X^*, Z^*))$ **then**
44: $\quad\quad\quad\quad\quad (X^*, Z^*) \leftarrow (X_k, Z_k)$
return (X^*, Z^*)

Fonte: (RAHIM et al., 2014)

6 Variantes propostas para o algoritmo de Rahim

O algoritmo de Rahim, descrito na Seção 5, propõe uma solução viável para o SIRP utilizando a relaxação lagrangeana. Nesse algoritmo, o Algoritmo 2 é utilizado para atualizar o LS da solução. Conforme descrito na Subseção 5.4.2, em cada iteração, o algoritmo resolve o problema de roteamento de veículo para cada período utilizando um método heurístico de economias que também verifica que as Restrições (19) e (21) não sejam violadas.

Com o objetivo de melhorar a solução encontrada, este trabalho apresenta três variações do algoritmo do Rahim, chamadas de Rahim com Centroide (Ct), Rahim com *Simulated Annealing* (SA) e Rahim com *Monte Carlo Savings* (MC). Nessas variações, é trocado o método heurístico baseado em economias do Algoritmo 2 por outros métodos conhecidos na literatura a saber: o algoritmo *baseado em Centroide* (SHIN; HAN, 2012), o algoritmo *Simulated Annealing* (HARMANANI et al., 2011) e o algoritmo *Monte Carlo Savings* (OLIVEIRA; DELGADO; MOREIRA, 2016), respectivamente. Esses algoritmos foram previamente introduzidos na seção 3.1. Cada um desses algoritmos foi implementado e adaptado para verificar que as Restrições (19) e (21) não sejam violadas e são descritos a seguir.

6.1 Algoritmo Simulated Annealing

O método de Simulated Annealing escolhe uma solução vizinha de maneira aleatória, como descrito na Seção 3.1.2. Na adaptação realizada para esse projeto são utilizados dois processos de transformações a cada iteração, *mover* e *substituir a média mais alta*.

A transformação *mover* encontra e fixa cinco pares de clientes que tenham as menores distâncias entre si incluindo o depósito. Ao fixar esses pares, a transformação não irá alterá-los. Em seguida, são selecionados outros cinco clientes aleatórios. Estes clientes são removidos de suas rotas e inseridos em rotas aleatórias, caso as restrições não sejam violadas.

A transformação *substituir a média mais alta* calcula a distância média de todos os pares de clientes, seleciona os cinco que possuem maior distância média e os remove de suas

rotas. Em seguida, seleciona outras cinco rotas aleatórias e insere os clientes que foram selecionados anteriormente em uma rota que resulte em um menor custo de atendimento.

Esse tipo de solução é sempre viável pois as transformações usam uma abordagem que respeita as restrições do problema. Se o custo da configuração vizinha (explorada na iteração) for melhor que a da melhor solução encontrada até o momento, ela se torna a nova melhor solução. De modo que, o algoritmo continua até que o tempo máximo de processamento $MaxTime$ seja alcançado ou até que o valor do parâmetro de temperatura T usado no algoritmo seja muito baixo.

O Algoritmo 4 apresenta o passo a passo dessa adaptação. Primeiramente é criado o conjunto de clientes SR_h (Linhas 1 – 8) que, assim como no Algoritmo 2, identificam os clientes que tem uma entrega no período h.

Em seguida, são instanciados os parâmetros básicos desse algoritmo (Linhas 9 – 19) que são α, β, M_0, T, $Time$ e $MaxTime$. Esses parâmetros influenciam na condição de parada do algoritmo. O parâmetro α é a taxa de decrescimento da temperatura T, M_0 é a unidade de tempo adicionado ao contador $Time$ a cada iteração até alcançar seu limite $MaxTime$ e β representa a taxa de crescimento de M_0 a cada iteração.

Por fim, no laço das Linhas 20–40, uma nova solução é investigada como candidata a melhor solução. Esse laço é executado enquanto T seja maior que 0.001 e o parâmetro $Time$ seja menor que um valor máximo estabelecido ($MaxTime$). Essa nova solução é selecionada aleatoriamente do conjunto de vizinhos da solução atual, em que cada vizinho é gerado com as operações de *mover* e *substituir a média mais alta* de maneira a respeitar as restrições do modelo, incluindo as restrições (19) e (21) (Linha 23). Caso o custo dessa nova solução seja menor que a melhor solução encontrada, então a nova solução é considerada como a melhor solução (Linhas 27–31). Caso contrário, a nova solução pode ser mantida como solução atual com uma certa probabilidade (Linhas 33–35). Por fim, os parâmetros são atualizados nas Linhas 37–39.

Finalmente, a melhor solução encontrada para o período h é adicionada na solução (X_k, Z_k) (Linha 41).

Algoritmo 4 EncontrarSoluçãoViávelcomSimulatedAnnealing

Require: $(X, Z)_I, (X, Z)_R, instancia$

1: $(X_k, Z_k) \leftarrow nil$
2: $(X^*, Z^*) \leftarrow (X, Z)_I$
3: **for all** $h \in H$ **do**
4: $\quad SR_h \leftarrow \{\}$
5: $\quad$ **for all** $j \in S$ **do**
6: $\quad\quad tempoTrajetodeposito \leftarrow tempo(deposito, j) + tempo(j, deposito)$
7: $\quad\quad$ **if** $(X^*, Z^*).q_{jh} > 0 \ \wedge tempoTrajetodeposito < \tau_h$ **then**
8: $\quad\quad\quad SR_h \leftarrow SR_h \cup \{j\}$
9: $\quad S_0 \leftarrow$ SoluçãoInicial(SR_h)
10: $\quad \alpha \leftarrow 0.99$
11: $\quad \beta \leftarrow 1.05$
12: $\quad M_0 \leftarrow 5$
13: $\quad bestS \leftarrow S_0$
14: $\quad T \leftarrow 5000$
15: $\quad atualS \leftarrow S_0$
16: $\quad atualCusto \leftarrow custo(atualS)$
17: $\quad bestCusto \leftarrow custo(bestS)$
18: $\quad Time \leftarrow 0$
19: $\quad MaxTime \leftarrow 100000000000$
20: $\quad$ **do**
21: $\quad\quad M \leftarrow M_0$
22: $\quad\quad$ **do**
23: $\quad\quad\quad novoS \leftarrow Vizinho(atualS, instancia)$
24: $\quad\quad\quad novoCusto \leftarrow custo(novoS)$
25: $\quad\quad\quad \Delta_{custo} \leftarrow novoCusto - atualCusto$
26: $\quad\quad\quad$ **if** $\Delta_{custo} < 0$ **then**
27: $\quad\quad\quad\quad atualS \leftarrow novoS$
28: $\quad\quad\quad\quad atualCusto \leftarrow custo(atualS)$
29: $\quad\quad\quad\quad$ **if** $novoCusto < bestCusto$ **then**
30: $\quad\quad\quad\quad\quad bestS \leftarrow novoS$
31: $\quad\quad\quad\quad\quad bestCusto \leftarrow custo(bestS)$
32: $\quad\quad\quad\quad$ **else**
33: $\quad\quad\quad\quad\quad$ **if** $Random < e^{-\frac{\Delta_{custo}}{T}}$ **then**
34: $\quad\quad\quad\quad\quad\quad atualS \leftarrow novoS$
35: $\quad\quad\quad\quad\quad\quad atualCusto \leftarrow custo(atualS)$
36: $\quad\quad$ **while** $(M \geq 0)$
37: $\quad\quad Time \leftarrow Time + M_0$
38: $\quad\quad T \leftarrow \alpha * T$
39: $\quad\quad M_0 \leftarrow \beta * M_0$
40: $\quad$ **while** $(Time < MaxTime \ \& \ T > 0.001)$
41: $\quad (X_k, Z_k).addRotaXZ(bestS)$
return (X_k, Z_k)

6.2 Algoritmo Baseado em Centroide

O método adaptado de (SHIN; HAN, 2012) consiste em três etapas de construção da solução. Na primeira fase, são construídos grupos (clusters) de clientes. Cada grupo é construído selecionando o cliente mais distante do depósito como semente do grupo. Após isso, são selecionados os clientes mais próximos do centroide do cluster, sempre atualizando a localização do centroide ao adicionar um novo cliente. Esse processo continua até a capacidade do veículo não ser ultrapassada. Quando isso acontece, a construção do grupo é finalizada, uma nova semente é selecionada e o processo de construção de um novo grupo começa novamente até que todos os clientes estejam em um grupo.

A segunda fase, ajusta os grupos formados com auxílio do centro geométrico dos grupos. Essa etapa verifica se um cliente está mais próximo do centroide de um grupo vizinho do que o centroide de seu grupo atual. Caso isso aconteça e a capacidade do veículo do grupo vizinho não seja violada, o cliente é movido para o grupo vizinho e os centroides são recalculados. Na última fase, as rotas são estabelecidas para o atendimento dos clientes de cada grupo. Para isso, a heurística apresentada em (LIN; KERNIGHAN, 1973) é usada para resolver o TSP.

O Algoritmo 5 apresenta o passo a passo dessa adaptação. Para cada período h de planejamento, o conjunto SR_h de clientes que tem uma entrega a ser realizada é construído (Linhas 1 – 8). Em cada período os clientes selecionados são divididos em grupos que respeitam as restrições do problema, incluindo as restrições (19) e (21), e essa solução inicial é considerada como a melhor solução até o momento (Linhas 9–11).

Cada iteração no laço das Linhas 13–18 do algoritmo realiza o processo de ajuste dos agrupamentos, considerando a distancia de cada cliente com o centroide dos grupos. Em cada iteração, após o ajuste realizado, é encontrada a melhor rota para cada grupo que respeitem as restrições do modelo, resolvendo o problema de TSP (Linha 15) e se o custo dessa solução atual, considerando o ajuste, for menor que a melhor solução encontrada, a solução atual se torna a melhor solução do algoritmo (Linhas 17–18). O laço irá continuar até que não for possível realizar ajustes nos grupos.

Por fim, a melhor solução encontrada é adicionada na solução (X_k, Z_k) como a solução para o período h (Linha 19).

Algoritmo 5 EncontrarSoluçãoViávelcomAlgoritmoCentroide

Require: $(X, Z)_I, (X, Z)_R, instancia$

$(X_k, Z_k) \leftarrow nil$
$(X^*, Z^*) \leftarrow (X, Z)_I$
for all $h \in H$ **do**
    $SR_h \leftarrow \{\}$
    for all $j \in S$ **do**
        $tempoTrajetodeposito \leftarrow tempo(deposito, j) + tempo(j, deposito)$
        if $(X^*, Z^*).q_{jh} > 0 \ \wedge tempoTrajetodeposito < \tau_h$ **then**
            $SR_h \leftarrow SR_h \cup \{j\}$
    $Clusters \leftarrow$ Construção_Inicial_Cluster_proximidade(SR_h)
    Adiciona o centro de distribuição aos $Clusters$
    melhor_solução $\leftarrow$ Resolve o problema de TSP($Clusters$)
    Remove o centro de distribuição dos $Clusters$
    while Clusters são ajustados **do**
        Adiciona o centro de distribuição aos $Clusters$
        atual_solução $\leftarrow$ Resolve o problema de TSP($Clusters$)
        Remove o centro de distribuição dos $Clusters$
        if custo(atual_solução) $<$ custo(melhor_solução) **then**
            melhor_solução $\leftarrow$ atual_solução
    (X_k, Z_k).addRotaXZ(melhor_solução)
    return (X_k, Z_k)

6.3 Algoritmo Monte Carlo Savings

O algoritmo que combina dois dos métodos mais conceituados é apresentado em (OLIVEIRA; DELGADO; MOREIRA, 2016). As técnicas que são conciliadas nesse método para encontrar uma solução são o método Clarke & Wright e simulações de Monte Carlo.

O método C&W consiste em construir uma matriz de economias, em que cada elemento dessa matriz representa a economia de se atender um par de clientes pelos veículos em operação. O valor dessa matriz é normalmente obtido por uma função de economia como a exemplificada na Seção 3.1.1. Após a matriz ser construída, o método a ordena de forma descrescente. Em seguida, é iniciada a construção da solução inicial, criando uma rota para cada cliente do problema. Assim, o método inicia operações de combinação para juntar duas rotas considerando a ordem da matriz de economias. Para tal, dada uma economia s_{ij}, é determinado se existem duas rotas que possam ser combinadas por s_{ij}, sendo necessário que a primeira rota termine no cliente i e a segunda rota tenha como primeiro cliente o cliente j.

O método denominado de *Monte Carlo Savings* utiliza os mesmos passos do C&W para encontrar uma solução do problema. Porém, são realizas r simulações que alteram a ordem da matriz de economias, adicionando um peso a uma determinada economia s_{ij} de maneira aleatória. Dessa forma, o método adaptado de C&W é executado r vezes com diferentes ordens em sua matriz e o método armazena dentre as r simulações aquela que obteve o menor custo de solução.

Algoritmo 6 EncontrarSoluçãoViávelcomMonteCarloSavings

Require: número de simulações r, limite do intervalo aleatório λ, $(X,Z)_I, (X,Z)_R, instancia$

1: $(X_k, Z_k) \leftarrow nil$
2: $(X^*, Z^*) \leftarrow (X,Z)_I$
3: **for all** $h \in H$ **do**
4: $\quad SR_h \leftarrow \{\}$
5: $\quad$ **for all** $j \in S$ **do**
6: $\quad\quad tempoTrajetodeposito \leftarrow tempo(deposito, j) + tempo(j, deposito)$
7: $\quad\quad$ **if** $(X^*, Z^*).q_{jh} > 0 \ \wedge tempoTrajetodeposito < \tau_h$ **then**
8: $\quad\quad\quad SR_h \leftarrow SR_h \cup \{j\}$
9: $\quad best \leftarrow$ create_initial_routes(SR_h)
10: $\quad$ **for** $k = 0$ to r **do**
11: $\quad\quad solution \leftarrow$ create_initial_routes(SR_h)
12: $\quad\quad saving\_list \leftarrow$ COMPUTE_MCS_SAVINGS(SR_h,λ)
13: $\quad\quad$ **for** $i, j \in saving\_list$ **do**
14: $\quad\quad\quad$ **if** $feasible\_merge(i, j, k_v)$ **then**
15: $\quad\quad\quad\quad solution \leftarrow merge\_routes(i, j)$
16: $\quad\quad$ **if** $custo(solution) < custo(best)$ **then**
17: $\quad\quad\quad best \leftarrow solution$
18: $\quad (X_k, Z_k)$.addRotaXZ(best)
return (X_k, Z_k)

Algoritmo 7 COMPUTE_MCS_SAVINGS

Require: limite do intervalo aleatório λ, $instancia$

1: $saving\_list \leftarrow \{\}$
2: **for** $i, j \in S - 0$ **do**
3: $\quad s \leftarrow custo(0, i) + custo(0, j) - custo(i, j)$
4: $\quad p \leftarrow random(-\lambda, +\lambda)$
5: $\quad S_{i,j} \leftarrow s + (s * p)$
6: $\quad savings\_list \leftarrow S_{i,j}$
7: $ordena\_descrecente(savings\_list)$
return saving_list

Os Algoritmos 6 e 7 detalham o método de *Monte Carlo Savings*. O Algoritmo 7 é responsável por variar a ordem da matriz de forma que para cada par de clientes i, j (Linha

2) é calculada a economia desse par pela equação $s \leftarrow custo(0,i) + custo(0,j) - custo(i,j)$ (Linha 3). Em seguida, é calculado o peso que será atribuído a essa economia. Esse peso é então incorporado a economia S_{ij} que será adicionado a matriz de economias (Linhas 4–6). E por fim, após realizar esse procedimento para todos os pares, a matriz é ordenada de forma descrescente (Linha 7).

O Algoritmo 6 primeiro identifica o grupo de clientes SR_h para cada período h de planejamento (Linhas 1 – 8). Em seguida, cria a solução inicial considerando os clientes identificados (Linha 9), em que cada cliente é atendido por uma rota diferente. Diante disso, para cada simulação r, uma nova matriz de economias é gerada pelo método COMPUTE_MCS_SAVINGS chamado na Linha 12. A nova matriz é percorrida do inicio ao fim (Linhas 13–15), e para cada par i, j encontrado é verificada a possibilidade de combinação. Uma combinação só é valida se respeitar as restrições do modelo, inclusive as restrições (19) e (21). Nesse caso, a solução atual é atualizada (Linha 15). Caso essa solução tenha o custo menor que a melhor solução encontrada até o momento, a solução atual se torna a melhor solução (Linhas 16–17). A solução final do período h é então adicionada na solução (X_k, Z_k)(Linha 18).

7 Modelo proposto para SIRPTW

Neste capítulo é apresentado um novo modelo para SIRP com janelas de tempo (Stochastic Inventory Routing Problem with Time Window – SIRPTW). Com a adição de novas restrições, conseguimos atender outras características que são frequentemente encontradas na realidade, de modo que este modelo obtém maior flexibilidade e aplicabilidade.

7.1 Modelagem estocástica

As variáveis e parâmetros para modelar o SIRPTW são os mesmos usados em (RAHIM et al., 2014) (descritos no Capitulo 5) mais as seguintes variáveis de decisão:

- t_{jhv}: horário de chegada e inicio de atendimento ao cliente $j \in S$ no período $h \in H$ pelo veículo $v \in V$;
- t^d_{jhv}: horário de saída do cliente $j \in S$ no período $h \in H$ pelo veículo $v \in V$;

e os seguintes parâmetros:

- C_j: capacidade de estoque do cliente $j \in S$;
- b_{jh}: horário de início da janela de atendimento do cliente $j \in S$ no período $h \in H$;
- e_{jh}: horário de fim da janela de atendimento do cliente $j \in S$ no período $h \in H$;
- tq_{jh}: tempo de entrega do cliente $j \in S$ no período $h \in H$;

O modelo de programação inteira mista e estocástico (PIME) proposto para o problema SIRPTW neste trabalho é:

$$min \sum_{h\in H}\sum_{v\in V}\left(\psi_v y_{hv} + \sum_{(i,j)\in S^2}\Gamma_{ijhv}x_{ijhv}\right) + \sum_{h\in H^+}\sum_{j\in S}\eta_{jh}I_{jh} \tag{38}$$

$s.a.:$

$$\sum_{v\in V}\sum_{i\in S} x_{ijhv} \leq 1, \quad \forall j \in \zeta, h \in H, \tag{39}$$

$$\sum_{i\in S}(x_{ijhv} - x_{jihv}) = 0, \quad \forall j \in S, h \in H, v \in V, \tag{40}$$

$$\sum_{(i,j)\in S^2} \theta_{ij}x_{ijhv} \leq \tau_h, \quad \forall h \in H, v \in V, \tag{41}$$

$$\sum_{v\in V}\sum_{i\in S}(Q_{ijhv} - Q_{jihv}) = q_{jh}, \quad \forall j \in \zeta, h \in H, \tag{42}$$

$$Q_{jkhv} \leq k_v x_{jkhv}, \quad \forall i, j \in S, h \in H, v \in V, \tag{43}$$

$$I_{r,h-1} + R_{rh} - I_{r,h} = \sum_{j\in\zeta} q_{jh}, \quad \forall j \in \zeta, h \in H, \tag{44}$$

$$I_{j,h-1} + q_{jh} - I_{j,h} = \Lambda_{jh}, \quad \forall j \in \zeta, h \in H, \tag{45}$$

$$I_{j0} \leq I_{jT}, \quad \forall j \in \zeta, \tag{46}$$

$$x_{rjhv} \leq y_{hv} \quad \forall j \in \zeta, h \in H, v \in V, \tag{47}$$

$$I_{j,h-1} + q_{jh} \leq C_j, \quad \forall j \in \zeta, h \in H, \tag{48}$$

$$t_{jhv} \geq t^d_{ihv} + \theta_{ij} - M(1 - x_{ijhv}), \quad \forall i \in \zeta, j \in \zeta, v \in V, h \in H \tag{49}$$

$$\sum_{i\in\zeta} x_{ijhv}b_{jh} \leq t_{jhv} \leq \sum_{i\in\zeta} x_{ijv}e_{jh}, \quad \forall j \in \zeta, v \in V, h \in H, \tag{50}$$

$$t^d_{0hv} = 0, \quad \forall v \in V, h \in H, \tag{51}$$

$$\sum_{i\in\zeta} x_{ijhv}t^d_{ihv} = \sum_{i\in\zeta} x_{ijhv}(t_{ihv} + tq_{ih}), \quad \forall j \in \zeta, v \in V, h \in H, \tag{52}$$

$$t_{ihv} \geq 0, t^d_{ihv} \geq 0, \quad \forall i \in \zeta, v \in V, h \in H, \tag{53}$$

$$I \geq 0, q \geq 0, \tag{54}$$

$$x, y \in \{0,1\}^{|S|^2|H||V|}, Q \geq 0 \tag{55}$$

em que a função objetivo (38) minimiza três custos operacionais que devem ser considerados no modelo. O primeiro é o custo de operação do veículo que consiste no valor gasto pelo proprietário da frota no momento em que o veículo, que realizará a operação, sai do seu estado de repouso. Esse custo normalmente é constituído pela manutenção do veículo e mão de obra da equipe encarregada.

O segundo custo é o custo de transporte e de entrega. Esse custo é associado a todo gasto que é obtido durante o trajeto de operação, que pode conter desde a depreciação do veículo, combustível e desgaste dos pneus.

Por fim, o terceiro custo é a despesa relacionada ao armazenamento do produto. Esse custo pode ser descrito como os dispêndios necessários para manter a estrutura e condições para que o fornecedor armazene e gerencie todos os produtos que estão em estoque. Dentro desses, podem ser considerados o aluguel do local de estocagem, a aquisição de equipamentos para manipulação dos produtos e a contratação de pessoal responsável pelo gerenciamento do armazém. No modelo, esse custo é considerado somente no final de cada período para que seja incluída toda a movimentação de estoque realizada pela operação.

Além disso, temos critérios que definem as características e restrições do modelo proposto. A partir da identificação de fatores comuns nesses critérios foi possível agrupá-los. Os critérios que atuam nos aspectos relacionado ao transporte são (39, 40, 41, 43 e 47). As restrições (44, 45, 46 e 48) tem em comum a influência sob o estoque, tanto do cliente quanto do depósito. A restrição (42) é a única restrição que une os dois aspectos do modelo, as restrições (54) e (55) representam o domínio das variáveis básicas do modelo. E por fim, como último grupo, os critérios relacionados a janela de tempo são (49–53).

A primeira restrição relacionada aos critérios de transporte é (39). Ela garante que cada cliente seja visitado no máximo uma vez em um único período, evitando custos adicionais de transporte. A Restrição (40) assegura que a rota de um veículo deve sempre começar no depósito, passar pelos clientes e retornar para o depósito. De modo que seja estabelecido um ciclo de rota. Já a Restrição (41) garante que todo o tempo de trajeto da rota do veículo não ultrapasse as horas de trabalho do período. Essa restrição é importante pois com sua aplicação, o fornecedor da operação pode programar a jornada de trabalho da sua equipe. A Restrição (43) assegura que a capacidade do veículo não vai ser ultrapassada e que as variáveis Q_{ijhv} não possam transportar qualquer quantidade de produto a menos que x_{ijhv} tenha valor 1. A Restrição (47) garante que um cliente não pode ser atendido por um veículo, quando esse não estiver em operação no período. Essa é a restrição que relaciona a utilização do veículo com o atendimento dos clientes.

A primeira restrição associada ao estoque é a (44), que realiza o nivelamento do estoque do depósito, considerando a política de reabastecimento do produto no estoque do depósito no período. A Restrição (45) calcula a quantidade de entrega e o nível de

estoque de cada período considerando a demanda estocástica de cada cliente. A Restrição (46) assegura que o estoque no final do planejamento de um determinado cliente seja tão grande quanto o do início do planejamento, uma vez que o inventário no início do planejamento não é considerado na função objetivo deste modelo. Finalmente, a Restrição (48) foi incluída para atender a capacidade de estocagem dos clientes (C_j), ou seja, a entrega realizada somada ao estoque inicial de cada período não deve ser maior do que a capacidade máxima de cada cliente.

A Restrição (42), que une o aspecto de estoque e de transporte do modelo, determina a quantidade que sobra no veículo a cada entrega. Todas as restrições anteriores são similares às restrições do trabalho de Rahim et al. (2014), exceto a Restrição (48).

Adicionalmente, um conjunto de restrições foi inserido no modelo para tratar o problema de janela de atendimento. As restrições incluídas no modelo foram adaptadas do trabalho de Li et al. (2014) que resolvem IRPTW, previamente descritas na Seção 4.5.2. Foi necessário especificamente adaptar as restrições (9–14) pois no modelo original para IRPTW é considerado apenas um único período de planejamento. A Restrição (49) tem como objetivo eliminar subrotas derivadas do tempo de trajeto de cada rota, para isso é considerado um valor muito grande M. A Restrição (50) especifica que o horário, no qual o veículo irá chegar no cliente j, estará dentro da janela de atendimento definida. A Restrição (51) define o horário que o veículo irá sair do centro de distribuição e a Restrição (52) define o horário que o veículo irá sair de cada cliente. Por fim, a Restrição (53) define a integridade do horário de início e fim dos atendimentos de cada período.

7.2 Aproximação Determinística

Após a definição do modelo PIME para resolver o SIRPTW, foi realizada uma aproximação determinística similar a proposta por Rahim et al. (2014). Essa aproximação deve satisfazer a demanda estocástica em cada período com um determinado nível de confiança $(1-\alpha)$. Utilizando a desigualdade de Chebyshev (KVANLI; PAVUR; KEELING, 2005) a restrição (45) foi substituída por duas novas restrições que devem garantir que:

$$Probabilidade\left\{I_{j,h-1}+\sum_{s=h}^{T}q_{jh}\geq\sum_{s=h}^{T}d_{js}\tau_s\right\}=(1-\alpha_{jh}),\quad\forall j\in S,h\in H$$

As novas restrições são definidas como:

$$I_{j,h-1} + \sum_{s=h}^{T} q_{js} \geq \sum_{s=h}^{T} E\left(d_{js}\right)\tau_s + z^{\alpha_{jh}}\left(\sqrt{T-h+1}\right)\sigma_j, \quad \forall j \in \zeta, h \in H, \tag{56}$$

$$E\left(d_{j1}\right)\tau_1 + z^{\alpha_{j1}}\sigma_j - I_{j0} \leq \sum_{v\in V}\sum_{i\in S} Q_{ij1v}, \quad \forall j \in \zeta \tag{57}$$

Essas restrições (a primeira relacionada com a parte de estoque e a segunda com transporte) levam em consideração o nível de confiança para a estocasticidade da demanda dos clientes. Note que no modelo Rahim et al. (2014), esse nível de confiança é único para todos os clientes, porém em um cenário real é necessário um nível de confiança individualizado para cada cliente. O nível de confiança individualizado permite dar prioridade a alguns clientes específicos. Por exemplo, no ramo de entrega de gás industrial, é necessário que o nível de confiança de um hospital seja maior do que o nível de confiança de um restaurante.

Assim, a nova Restrição (56) previne a escassez de estoque em cada cliente j com um nível de confiança $(1-\alpha_{jh})$. Assim, $100*(1-\alpha_{jh})\%$ do nível de serviço é garantido para cada cliente $j \in \zeta$ em cada período $h \in H$, definido pelo valor normal padrão $z^{\alpha_{jh}}$. Enquanto que a Restrição (57) garante com certo nível de confiança que não ocorrerá escassez em cada cliente durante o primeiro período no horizonte de planejamento. Incluindo essas novas restrições, o modelo determinístico para o SIRPTW é:

$$min \sum_{h\in H}\sum_{v\in V}\left(\psi_v y_{hv} + \sum_{(i,j)\in S^2}\Gamma_{ijhv}x_{ijhv}\right) + \sum_{h\in H^+}\sum_{j\in S}\eta_{jh}I_{jh} \tag{58}$$

$$s.a: (39)-(44), (46)-(57).$$

7.3 Usando Relaxação Lagrangeana

Assim como no modelo de (RAHIM et al., 2014), a relaxação lagrangeana é aplicada a uma restrição do problema considerada como complicadora. Nesse modelo foi selecionada a restrição (42) pois ela é a única que combina variáveis de estoque e transporte. Essa restrição foi eliminada e inserida na função objetivo com multiplicadores lagrangeanos μ_{jh} para todo $j \in S, h \in H$, obtendo:

$$min \sum_{h\in H}\sum_{v\in V}\left(\psi_v y_{hv} + \sum_{(i,j)\in S^2}\Gamma_{ijhv}x_{ijhv}\right) + \sum_{h\in H^+}\sum_{j\in S}\eta_{jh}I_{jh}$$
$$+ \sum_{h\in H}\sum_{j\in\zeta}\mu_{jh}\left(q_{jh} - \sum_{v\in V}\sum_{i\in S}(Q_{ijhv} - Q_{jihv})\right) \quad (59)$$
$$s.a : (39) - (41), (43), (44), (46) - (57).$$

Agora que a restrição (42) foi eliminada, e assim é possível decompor o modelo em dois submodelos. O primeiro atua sobre os aspectos de estoque, sendo esse:

$$min \sum_{h\in H^+}\sum_{j\in S}\eta_{jh}I_{jh} + \sum_{h\in H}\sum_{j\in\zeta}\mu_{jh}q_{jh} \quad (60)$$
$$s.a : (44), (46), (48), (56)\, e\ (54).$$

O segundo submodelo, responsável pelos aspectos de transporte (roteamento) e janelas de tempo, é definido como:

$$min \sum_{h\in H}\sum_{v\in V}\left(\psi_v y_{hv} + \sum_{(i,j)\in S^2}\Gamma_{ijhv}x_{ijhv}\right) -$$
$$\sum_{h\in H}\sum_{j\in\zeta}\mu_{jh}\left[\sum_{v\in V}\sum_{i\in S}(Q_{ijhv} - Q_{jihv})\right] \quad (61)$$
$$s.a : (39) - (41), (43), (47), (49) - (53), (57)\, e\ (55).$$

O melhor limitante inferior para o Modelo (58) pode ser calculado a partir do vetor de multiplicadores lagrangeano ótimo que é a solução do dual do Modelo (59), a seguir:

$$max \quad LMP(\mu_{jt}), \quad (62)$$

em que $LMP(\mu_{jt})$ é:

$$min \sum_{h\in H}\sum_{v\in V}\left(\psi_v y_{hv} + \sum_{(i,j)\in S^2}\Gamma_{ijhv}x_{ijhv}\right) + \sum_{h\in H^+}\sum_{j\in S}\eta_{jh}I_{jh}$$
$$+ \sum_{h\in H}\sum_{j\in\zeta}\mu_{jh}\left[q_{jh} - \sum_{v\in V}\sum_{i\in S}(Q_{ijhv} - Q_{jihv})\right] \quad (63)$$

7.4 Resolvendo o problema dual

A solução para o problema dual segue a ideia proposta por Rahim et al. (2014). É utilizado o algoritmo de subgradiente que ajusta os multiplicadores lagrangeanos μ_{jt} gerando limitantes superiores e inferiores a cada iteração do algoritmo. Para ajustar o limitante inferior os subproblemas de inventário (Modelo 60) e o subproblema de roteamento e janelas de tempo (Modelo 61) apresentados na Seção 7.3 são solucionados. Em seguida, uma heurística que recebe as soluções dos subproblemas é aplicada para ajustar o limitante superior, i.e., para encontrar uma solução viável.

Em termos de implementação, ao incluir as restrições de janela de tempo ao modelo de (RAHIM et al., 2014), foi necessário adicionar nos Algoritmos 1 e 3 (apresentados no Capítulo 5) as Restrições (49)-(53) para verificar se a solução é viável. Essa adaptação não afetou a estrutura desses algoritmos, somente suas funções de validação de uma solução viável. Já o Algoritmo 2, que implementa uma heurística para encontrar uma solução viável proposta por Rahim et al. (2014), quanto as que foram apresentadas no Capitulo 6 não podem ser usadas pois elas lidam apenas com problemas de roteamento simples, i.e. não incluem janelas de tempo. Dessa forma, foi necessário implementar outros métodos para solucionar esse problema.

7.5 Algoritmos heurísticos para encontrar uma solução viável

Os algoritmos escolhidos para serem adaptados para encontrar uma solução viável do SIRPTW foram os propostos em (SOLOMON, 1987) e (TAN et al., 2001). Esses algoritmos adaptam os método de Clarke & Wright e Simulated Annealing, respectivamente, para o problema de roteamento com janela de tempo (descritos na Seção 3.2). O motivo pelo qual selecionamos esses métodos se deve aos bons resultados obtidos por métodos semelhantes para resolver SIRP (ver Seção 8.2).

7.5.1 Clarke & Wright com janela de tempo

O método desenvolvido em (SOLOMON, 1987) adapta a heurística de economias de C&W de forma a atender o problema com janela de tempo. Para isso, é adicionada,

na combinação de duas rotas pelo par i, j, a verificação se essa combinação invalida a janela de todos os clientes que estão nas duas rotas. Os autores afirmam que somente essa adaptação no algoritmo pode gerar uma nova complicação com o método, pois, devido a função de economias, é possível que o método junte dois clientes muito próximos em distancia porém muito distantes considerando suas janelas. Para contornar esse problema foi adicionado também um tempo limite no tempo de espera de cada atendimento.

O Algoritmo 8 apresenta a adaptação do método C&W. Como explicado, o algoritmo é bem similar ao Algoritmo 2, sendo que a principal modificação é a inclusão da validação das restrições que respeitam as regras de janela e a verificação do tempo máximo de espera para inicio da operação (Linhas 27 e 28 do Algoritmo 8).

7.5.2 Simullated Annealing com janela de tempo

O Algoritmo 9 apresenta uma adaptação do método Simulated Annealing proposto em (TAN et al., 2001). O algoritmo começa por selecionar uma solução inicial (configuração de rotas para atender as entregas dos clientes) gerada pela heurística PFIH e em cada iteração utiliza a heurística $LSD(GB)$ para aprimorar uma solução existente. A seguir são descritas essas duas heurísticas:

- A heurística de inserção para frente (em inglês Push Forward Insertion Heuristic - PFIH) inicia uma nova rota com um dos clientes a ser atendido. Em seguida, o algoritmo tenta incluir um cliente por vez em cada aresta da rota, verificando a viabilidade da janela de tempo e capacidade, e calculando o custo dessa inserção. Quando a inserção de um cliente não é possível devido a violação das restrições, uma nova rota é criada e o processo de inclusão recomeça até que todos os clientes estejam em uma rota.
- O método 2-interchange LSD(GB) tem como objetivo melhorar uma solução através de operações de trocas entre clientes de rotas diferentes. Cada troca é definida como uma operação (p, q), no qual p clientes da rota R_p serão trocados com q clientes da rota R_q. Nesse método, o número máximo de clientes que podem ser trocados de uma única rota é dois. Cada troca é definida com o menor custo de inserção entre as arestas da rota destino respeitando as restrições de janela de tempo e capacidade máxima da rota.

Algoritmo 8 EncontrarSoluçãoViávelC&WcomJanelaTempo

Require: $(X, Z)_I, (X, Z)_R, instancia$

1: $(X_k, Z_k) \leftarrow nil$
2: $(X^*, Z^*) \leftarrow (X, Z)_I$
3: **for all** $h \in H$ **do**
4: $SR_h \leftarrow \{\}$ ▷ Inicializando a rota
5: $Rota \leftarrow nil$
6: **for all** $j \in S$ **do**
7: $tempoTrajetodeposito \leftarrow tempo(deposito, j) + tempo(j, deposito)$
8: **if** $(X^*, Z^*).q_{jh} > 0 \ \wedge tempoTrajetodeposito < \tau_h$ **then**
9: $SR_h \leftarrow SR_h \cup \{j\}$
10: $Tour \leftarrow \{j\}$
11: $Rota.add\,(Tour)$
12: $melhorEconomia \leftarrow -1$
13: **repeat** ▷ Combinando tours
14: $n \leftarrow tamanho(Rota)$
15: $melhorTouri \leftarrow -1$
16: $melhorTourj \leftarrow -1$
17: $melhorTour \leftarrow \{\}$
18: $TourC^+ \leftarrow \{\}$
19: **for** $i = 0$ to n **do**
20: **for** $j =$ i+1 to n **do**
21: $TourC^+ \leftarrow Rota[i] \cup Rota[j]$
22: TSP$(TourC^+)$
23: $tempoTour \leftarrow calculaTempoTrajeto(TourC^+)$
24: **if** $tempoTour > \tau_h \vee naoValidaRestricoes(TourC^+)$ **then**
25: $Desconsidera(i, j)$
26: **else**
27: **if** $naoValidaRestricoesJanelas(TourC^+)$
28: $\vee\ tempoEsperaInvalido(TourC^+)$ **then**
29: $Desconsidera(i, j)$
30: **else**
31: $tempoTrajetoi \leftarrow calculaTempoTrajeto(Rota[i])$
32: $tempoTrajetoj \leftarrow calculaTempoTrajeto(Rota[j])$
33: **if** $tempoTour < tempoTrajetoi + tempoTrajetoj$ **then**
34: $SV \leftarrow tempoTrajetoi + tempoTrajetoj - tempoTour$
35: **if** $SV > melhorEconomia$ **then**
36: $melhorEconomia \leftarrow SV$
37: $melhorTour \leftarrow TourC^+$
38: $melhorTouri \leftarrow i$
39: $melhorTourj \leftarrow j$
40: **if** $melhorTouri > -1 \wedge melhorTourj > -1$ **then**
41: $Rota.remove(melhorTouri)$
42: $Rota.remove(melhorTourj)$
43: $Rota.add(melhorTour)$
44: **until** $melhorEconomia > 0$
45: calculaCustoRota$(Rota)$
46: calculanúmeroVeículos$(Rota)$
47: $(X_k, Z_k).addRotaXZ(Rota)$
return (X_k, Z_k)

Algoritmo 9 EncontrarSoluçãoViávelSimulatedAnnealingcomJanelaTempo

Require: $(X, Z)_I$, $(X, Z)_R$, *instancia*, R

1: $(X_k, Z_k) \leftarrow nil$
2: $(X^*, Z^*) \leftarrow (X, Z)_I$
3: **for all** $h \in H$ **do**
4: $SR_h \leftarrow \{\}$
5: **for all** $j \in S$ **do**
6: $tempoTrajetodeposito \leftarrow tempo(deposito, j) + tempo(j, deposito)$
7: **if** $(X^*, Z^*).q_{jh} > 0 \ \wedge tempoTrajetodeposito < \tau_h$ **then**
8: $SR_h \leftarrow SR_h \cup \{j\}$
9: Inicializar S;S_b; R; τ;
10: Gerar solução inicial S aplicando o método *PFIH*
11: Aperfeiçoar S aplicando o método *2-interchange LSD(GB)*
12: Inicializa parâmetros de cozimento $T_b, T_r, T_k = 100$,$\tau = 0.5$
13: **while** $nReset < R$ **do**
14: Criar uma solução $S^{'} \in N_2(S)$
15: Calcular $\Delta = C(S^{'}) - C(S)$
16: $\theta =$ Aleatório(0,1)
17: **if** $(\Delta \leq 0)$ ou $(\Delta > 0$ e $exp(-\Delta/T_k) \geq \theta)$ **then**
18: $S = S^{'}$
19: **if** $C(S) < C(S_b)$ **then**
20: Aperfeiçoar S aplicando o método *2-interchange LSD(GB)*
21: $S_b = S$
22: $T_b = T_k$
23: $nReset = 0$
24: $k = k + 1$
25: $T_k = \frac{T_{k-1}}{1+\tau\sqrt{T_{k-1}}}$
26: **if** $N_2(S)$ não contém soluções válidas **then**
27: $T_r = max(T_r/2, T_b)$
28: $T_k = T_r$
29: $nReset = nReset + 1$
30: (X_k, Z_k).addRotaXZ(S_b)
return (X_k, Z_k)

No Algoritmo 9, primeiramente é criado o conjunto de clientes SR_h (Linhas 1 – 8) que tem uma entrega no período h. Considerando uma solução inicial S gerada pela heurística *PFIH* (Linha 10), o algoritmo utiliza a heurística *2-interchange LSD(GB)* para aprimorar uma solução existente (Linha 11).

Os parâmetros de cozimento T_b, T_r, T_k e τ são inicializados na Linha 12, sendo que T_b indica a temperatura em que a melhor solução foi encontrada, T_k é a representação da temperatura atual da iteração k e τ indica a taxa de diminuição da temperatura T_k. O parâmetro T_r representa a temperatura de recomeço calculada na Linha 27. Esse parâmetro

é recalculado no caso do algoritmo não encontrar soluções vizinhas válidas para a solução atual.

No laço das Linhas 13 – 29 primeiro é criada uma solução $S^{'}$ que pertence ao conjunto $N_2(S)$ de soluções vizinhas a S (Linha 14). O coeficiente Δ representa a diferença do custo entre as soluções $S^{'}$ e S (Linha 15). Em seguida, com base no coeficiente Δ, o algoritmo verifica se a solução atual é melhor do que a melhor solução encontrada até o momento S_b (Linhas 17 – 23).

O laço das Linhas 13 – 29 será executado até realizar um numero R de recomeços, em que um recomeço atualiza a temperatura atual do algoritmo e acontece na condição de não haver soluções vizinhas viáveis a solução atual S. No momento em que essa condição é alcançada, os cálculos da temperatura de recomeço T_r são realizados (Linha 27) e é adicionada uma unidade ao contador de recomeços $nReset$ (Linha 29). Por fim, a melhor solução encontrada é considerada como a solução do período h e é adicionada na solução (X_k, Z_k) (Linha 30).

8 Experimentos realizados

Os experimentos apresentados neste capítulo foram realizados com base no trabalho de Rahim et al. (2014), as instâncias sintéticas foram geradas com base na proposta de Yu, Chen e Chu (2008) e a instância real foi construída com base no conhecimento provido do convívio do autor próximo do ramo de distribuição.

8.1 Gerador de instâncias sintéticas

Não há um benchmark padrão de instâncias de testes para ser utilizado nos experimentos para o problema SIRP (ADELMAN, 2004; LOU; WU; XIAO, 2009; JUAN et al., 2014). Alguns autores criaram suas próprias instâncias entre eles (BERTAZZI et al., 2013; YU; CHU; CHEN, 2006; BINGLEI; SHI; JIAN, 2005; YU; CHEN; CHU, 2008), outros autores acabaram utilizando instâncias de outros problemas como VRP e IRP já existentes na literatura e as adaptaram para serem estocásticas como (SHUKLA; TIWARI; CEGLAREK, 2013; CÁCERES-CRUZ et al., 2012). Por esse motivo, as comparações entre técnicas desenvolvidas por diferentes autores acaba se tornando inviável e dificilmente praticada nesse ramo da literatura.

Neste trabalho, é criada uma ferramenta de geração de instâncias padronizada conforme a geração proposta por (YU; CHEN; CHU, 2008). Esse gerador de instâncias recebe um arquivo com os principais parâmetros do cenário, que são: (i) nome de identificação da instância; (ii) $posX$, vetor de posições no eixo x dos clientes; (iii) $posY$, vetor de posições no eixo y dos clientes; (iv) $d_{jh}Min$, valor mínimo possível da demanda $\forall j \in \zeta, h \in H$; (v) $d_{jh}Max$, valor máximo possível da demanda $\forall j \in \zeta, h \in H$; (vi) C, vetor com o nível máximo de estoque de cada cliente; (vii) N, T e $|V|$; I_{j0}, $\forall j \in \zeta$; $\tau_h, \forall h \in H$; v_υ, k_υ e δ_υ, $\forall \upsilon \in V$; η_{jh} e φ_{jh}, $\forall j \in \zeta, h \in H$.

Note que, o arquivo de parâmetros de cenário contém valores mínimos e máximos possíveis para a demanda de cada cliente em cada período. Baseados nesses valores, o gerador de instâncias gera de maneira aleatória um valor para cada período no horizonte de planejamento, para depois calcular $D_j = E\left(d_{jh}\right)$ e o desvio padrão σ_j. É realizado também o cálculo de outros atributos como a distância e tempo do trajeto entre cada par de clientes. Além disso, diferente do IRP determinístico não é possível determinar uma

Figura 2 – Gerador de instâncias

solução exata para uma instância devido à estocasticidade do SIRP. Porém, é possível determinar um limite inferior e um limite superior da solução, informação que pode ser usada para comparação entre técnicas distintas e que pode ser colocada no arquivo da instância.

Finalmente, um arquivo de instância em um formato padrão de texto é gerado. Esse arquivo contém o posicionamento dos clientes e do centro de distribuição; distâncias e tempos de trajeto; a média e desvio padrão da demanda por cliente; e LI e LS do custo da solução. A Figura 2 mostra o fluxo da geração do arquivo de instância. Ao receber os parâmetros do cenário o gerador irá gerar os valores desse cenário, em seguida, irá criar o arquivo. Esse arquivo pode ser utilizado como entrada de um solucionador para o problema SIRP, e assim, o autor desse solucionador poderá compartilhar os arquivos de instâncias dos seus experimentos.

8.2 Avaliação das variantes propostas para o algoritmo de Rahim em instâncias sintéticas

A seguir são detalhados as instâncias, os resultados e a discussão referente aos experimentos realizados com o algoritmo de Rahim e suas variações propostas neste trabalho para SIRP.

8.2.1 Benchmark de instâncias

Foram considerados três conjuntos de instâncias diferentes ($N = 15, T = 3$), ($N = 25, T = 3$) e ($N = 50, T = 3$), em que N é o número de clientes e T o tamanho do

horizonte de planejamento. Esses conjuntos foram criados pelo nosso gerador de instâncias utilizando os mesmos parâmetros que (RAHIM et al., 2014) e cada instância foi nomeada no formato (Ax-y-Tz), em que x é o total de clientes na instância, y é identificador de sequência e z representa o tamanho do horizonte de planejamento. O conjunto com N=15 foi criado considerando que a localização dos clientes é gerada randomicamente em um quadrado de 30 por 30 quilômetros e o centro de distribuição está localizado no meio do quadrado. A demanda d_{jh} é gerada randomicamente entre 1 e 3 toneladas por hora. O número de horas por período é $\tau_h = 8$ e o valor de z_α é $1,64$. O custo de armazenamento η_{jh} é gerado randomicamente entre $0,1$ e $0,15$ reais por tonelada e o custo de entrega ϕ_{jh} com valor de 25 reais é o mesmo para cada cliente. O custo operacional ψ_v dos veículos é de 50 reais e o custo de viagem δ_v é de 1 real por km/h. A frota tem veículos homogêneos com capacidade de 60 toneladas com uma velocidade média de 50 km/h.

Alguns parâmetros foram modificados para a criação das instâncias com N=25 e N=50. Os clientes são randomicamente posicionados em um quadrado de 100 por 100 e 200 por 200 quilômetros para N=25 e N=50, respectivamente. O centro de distribuição se mantém na posição central do quadrado e a demanda d_{jh} é gerada randomicamente entre 0.1 e 3 toneladas por hora. O custo de entrega ϕ_{jh} é gerado com valor de 10 reais para cada cliente e o custo operacional ψ_v dos veículos é de 30 reais. Os outros parâmetros mantém o mesmo valor dos conjuntos de 15 clientes.

8.2.2 Resultados

As Tabelas 5, 6 e 7 mostram os resultados obtidos para $(N = 15, T = 3)$, $(N = 25, T = 3)$ e $(N = 50, T = 3)$, respectivamente. Para cada instância das tabelas são apresentados o LI e LS para o algoritmo, o tempo computacional da solução em segundos $(CPU(s))$ e o $Gap = \frac{LS-LI}{LS} \times 100\%$. Nas tabelas, aqueles algoritmos que não terminaram de executar em até 5 horas estão marcados como *Não Terminou* (NT). Nesta seção, o algoritmo original de Rahim, o qual usa o algoritmo C&W, será chamado de CW. Nas Figuras 3 e 4 é mostrada a porcentagem de melhoria no limitante superior e no tempo dos algoritmos propostos com relação a CW, respectivamente.

O algoritmo do Rahim com centroide (Ct) obteve resultados insatisfatórios considerando o LS em comparação com os demais algoritmos. Na maioria dos testes, o custo

final obtido ficou maior que o esperado quando comparado com CW, em média Ct no LS foi 2%, 1% e 13% pior para as instâncias N=15, N=25 e N=50, respectivamente. Um ponto positivo para esse método é que ele obteve o menor tempo computacional médio nas instâncias de N=15 e N=50, sendo mais rápido em 42% e 6%, respectivamente, do que CW. Já para N=25, essa abordagem foi 34% mais devagar. Note que, esse método depende da dispersão da localização dos clientes. Assim, é provável que em instâncias em que é possível observar uma clara separação de grupos de clientes, o algoritmo Ct obtenha melhores resultados, o que parece ter acontecido nas instâncias A15-7-T-3 e A15-8-T-3.

O algoritmo Rahim com Simulated Annealing (SA) apresentou bons resultados quando comparado ao algoritmo CW. Para N=15 e N=25 o algoritmo SA encontra soluções com custo em média 0,4% e 1% maior que o algoritmo CW. Porém, são nas instâncias com N=50 que essa abordagem se destaca, obtendo os menores custos entre todos os algoritmos, em média 17% menores. O algoritmo SA é 464% mais devagar para N=15, 154% para N=25 e de apenas 10% para N=50.

Por último, os resultados de LS e LI para N=15 e N=25 usando o algoritmo do Rahim com Monte Carlo Savings (MC) foram muito parecidos com os do algoritmo de CW. Apenas algumas mínimas diferenças foram notadas nos resultados, sendo de 1% menor para N=15 e de 2% menor para N=25. O tempo de processamento computacional é a desvantagem desse método, entre todas as abordagens implementadas, o algoritmo MC apresentou o maior tempo computacional para terminar os experimentos para N=15 e N=25 e para as instâncias de N=50 este algoritmo não terminou em menos de 5 horas.

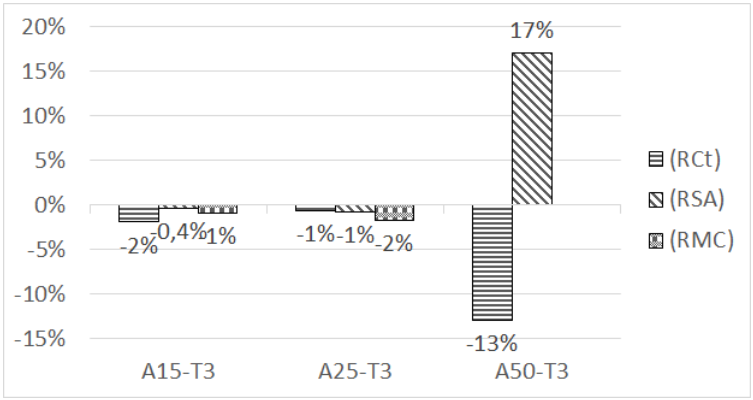

Figura 3 – Porcentagem de melhoria no limite superior com relação ao algoritmo CW

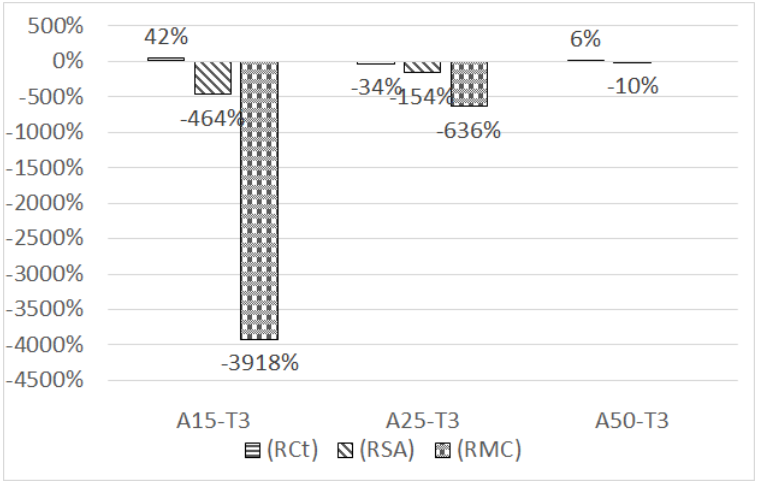

Figura 4 – Porcentagem de melhoria no tempo com relação ao algoritmo CW

Tabela 5 – Resultados das instâncias (N = 15, T = 3).

Instância	Limitante Inferior (CW)	(Ct)	(SA)	(MC)	Limitante Superior (CW)	(Ct)	(SA)	(MC)	Gap (%) (CW)	(Ct)	(SA)	(MC)	Tempo de CPU (s) (CW)	(Ct)	(SA)	(MC)
A15-0-T-3	**904**	902	**904**	**904**	**1080**	1122	**1080**	1122	**16,28**	19,64	**16,28**	19,43	**41**	59	249	4636
A15-1-T-3	**1387**	1378	1381	**1387**	**1749**	1760	**1749**	**1749**	20,73	21,73	21,07	**20,71**	121	**26**	274	5647
A15-2-T-3	1395	**1397**	1395	1395	**1663**	1700	**1663**	**1663**	**16,14**	17,83	**16,14**	**16,14**	**156**	167	1136	9775
A15-3-T-3	1442	**1443**	1442	1442	**1839**	1876	**1839**	**1839**	**21,58**	23,07	**21,58**	**21,58**	**104**	126	747	5888
A15-4-T-3	**1304**	**1304**	1302	**1304**	**1656**	1751	1706	1731	**21,28**	25,52	23,71	24,66	**33**	39	251	1787
A15-5-T-3	1337	**1338**	1337	1335	**1685**	1722	**1685**	**1685**	**20,69**	22,31	**20,69**	20,78	323	**27**	1669	2268
A15-6-T-3	1412	1401	1412	**1413**	**1779**	1798	**1779**	1784	**20,66**	22,10	**20,66**	20,83	191	**22**	812	7586
A15-7-T-3	1358	1357	**1359**	1358	1668	**1652**	1668	1668	18,57	**17,86**	18,52	18,57	**33**	35	300	1677
A15-8-T-3	**1379**	1362	**1379**	**1379**	1739	**1715**	1739	1739	20,69	**20,60**	20,69	20,69	32	**21**	247	1894
A15-9-T-3	954	**957**	955	956	**1001**	1062	1009	1015	**4,73**	9,86	5,33	5,74	**57**	109	471	2674
Média	**1287**	1284	1286	**1287**	**1586**	1616	1592	1600	**18,13**	20,05	18,47	18,91	109	**63**	616	4383

Tabela 6 – Resultados das instâncias (N = 25, T = 3).

Instância	Limitante Inferior (CW)	(Ct)	(SA)	(MC)	Limitante Superior (CW)	(Ct)	(SA)	(MC)	Gap(%) (CW)	(Ct)	(SA)	(MC)	Tempo de CPU (s) (CW)	(Ct)	(SA)	(MC)
A25-0-T-3	873	844	**877**	869	1162	1206	1162	**1144**	24,93	30,01	24,56	**23,99**	**525**	1061	1249	3329
A25-1-T-3	1132	**1134**	1125	1129	2013	**1633**	2090	2072	43,76	**30,56**	46,16	45,51	175	**164**	316	2794
A25-2-T-3	987	988	**991**	988	**1360**	1543	1388	1431	**27,43**	35,94	28,61	30,97	**194**	210	413	847
A25-3-T-3	**1140**	1139	**1140**	**1140**	1741	**1671**	1688	1688	34,50	**31,81**	32,48	32,48	269	**155**	492	1615
A25-4-T-3	**993**	**993**	**993**	992	1292	**1133**	1272	1238	23,14	**12,36**	21,92	19,88	**437**	938	897	1366
A25-5-T-3	1060	1061	**1064**	1058	1683	1793	**1491**	1829	37,05	40,80	**28,64**	42,15	**153**	231	474	4606
A25-6-T-3	999	999	1000	**1002**	**1140**	1422	1422	1219	**12,32**	29,72	29,69	17,79	363	**341**	784	1409
A25-7-T-3	1000	998	**1002**	1001	1582	1707	**1492**	1626	36,76	41,52	**32,83**	38,47	208	**198**	612	2017
A25-8-T-3	841	839	**842**	841	1347	**1171**	1425	1225	37,56	**28,36**	40,90	31,38	**386**	453	964	2487
A25-9-T-3	**945**	**945**	944	**945**	1395	1525	**1391**	1489	32,25	38,03	**32,12**	36,50	231	**201**	1273	1186
Média	997	994	**998**	996	**1472**	1481	1482	1496	**30,97**	31,91	31,79	31,91	**294**	395	747	2166

Tabela 7 – Resultados das instâncias (N = 50, T = 3).

Instância	Limitante Inferior (CW)	(Ct)	(SA)	(MC)	Limitante Superior (CW)	(Ct)	(SA)	(MC)	Gap (%) (CW)	(Ct)	(SA)	(MC)	Tempo de CPU (s) (CW)	(Ct)	(SA)	(MC)
A50-0-T-3	3497	3497	**3505**	NT	4882	5254	**4445**	NT	39,61	50,24	**26,82**	NT	**3677**	5656	5531	NT
A50-1-T-3	3534	3541	**3567**	NT	5050	5875	**4333**	NT	42,90	65,91	**21,47**	NT	5050	4870	**3774**	NT
A50-2-T-3	**3596**	3584	3568	NT	5540	6101	**4807**	NT	54,06	70,23	**34,73**	NT	**2852**	4321	6600	NT
A50-3-T-3	3345	**3366**	3357	NT	5276	5918	**4469**	NT	57,73	75,82	**33,12**	NT	2777	**2565**	2803	NT
A50-4-T-3	**3519**	**3519**	3516	NT	5409	5977	**4605**	NT	53,71	69,85	**30,97**	NT	4954	**4619**	7877	NT
A50-5-T-3	3555	3561	**3563**	NT	5800	6198	**4842**	NT	63,15	74,05	**35,90**	NT	3530	**1112**	3474	NT
A50-6-T-3	3528	**3552**	3520	NT	5228	6341	**4385**	NT	48,19	78,52	**24,57**	NT	6435	**3668**	5354	NT
A50-7-T-3	**4024**	4011	4015	NT	6098	6166	**4790**	NT	51,54	53,73	**19,30**	NT	6247	**6006**	6020	NT
A50-8-T-3	3456	**3480**	3446	NT	4925	6361	**4382**	NT	42,51	82,79	**27,16**	NT	**4193**	4296	5279	NT
A50-9-T-3	3725	3721	**3732**	NT	5088	5998	**4468**	NT	36,59	61,19	**19,72**	NT	3255	3252	**1426**	NT
Média	3578	**3583**	3579	NT	5330	6019	**4553**	NT	49,00	68,23	**27,38**	NT	4297	**4037**	4814	NT

8.2.3 Discussão

Dentre das variantes implementadas, todas apresentaram vantagens e desvantagens. Os experimentos mostraram que o algoritmo Rahim com Simulated Annealing é uma boa alternativa quando desejamos resolver instâncias grandes e complexas. Para instâncias menores, o melhor continua sendo o algoritmo original, i.e., o algoritmo Rahim com C&W. A redução do custo em comparação ao algoritmo de Rahim com C&W chega a 17% em média. Essa redução de custo se torna muito significativa quando analisamos a relevância do problema SIRP na operação logística.

Além disso, é possível observar que a diferença entre o tempo computacional para o algoritmo de Rahim com Simulated Annealing e do algoritmo de Rahim com C&W diminui enquanto aumentamos a quantidade de clientes, sendo o algoritmo Rahim com SA apenas 10% mais lento para N=50.

O GAP dos resultados confirma a escolha do algoritmo de Rahim com C&W para as instâncias menores e do algoritmo de Rahim com Simulated Annealing para instâncias maiores. Assim, mesmo que seja esperado o aumento do valor do GAP conforme o crescimento do tamanho das instâncias, os melhores GAPs acompanharam os melhores resultados.

8.3 Avaliação do algoritmo para o SIRPTW em instâncias sintéticas

A seguir são detalhados as instâncias, resultados e a discussão referente aos experimento realizados, considerando janela de tempo no problema de roteamento e inventário com demanda estocástica.

8.3.1 Benchmark de instâncias

Para avaliar o SIRPTW foi necessário gerar novas instâncias para o problema, cada instância foi nomeada no formato (Bx-y-Tz). A principal diferença entre as instâncias para o SIRPTW e as descritas na Seção 8.2.1 é que foram criadas novas variações de janelas de tempo para cada instâncias de 15, 25 e 50 clientes.

A primeira variação considera que todos os clientes de todas as instâncias tem uma janela restritiva (JR), considerando uma janela de tempo de no máximo 2 horas. A

segunda variação, considera uma janela mista (JM) no qual metade dos clientes tem janela restritiva e a outra metade tem a janela sem restrição, ou seja, a janela desses clientes considera todo o período de operação. E por último, a terceira variação considera todos os clientes com a janela sem restrição (JS).

Cada janela é gerada aleatoriamente em um período do dia entre 00:00 até 23:59. Quando falamos que uma janela é sem restrição, significa que sua janela inicia as 00:00 e termina as 23:59. Já as janelas restritivas tiveram seu início e fim gerados aleatoriamente nesse período.

8.3.2 Resultados

Nesta seção são comparados o método proposto nesse trabalho para resolver o SIRPTW utilizando a adaptação do método C&W, chamado de $CWTW$, e o método com a variação do Simulated Annealing para resolver o SIRTW, chamado de $SATW$. Os resultados obtidos para as instâncias de $(N = 15, T = 3)$, $(N = 25, T = 3)$ e $(N = 50, T = 3)$ resolvidas por esses dois métodos são apresentados nas Tabelas 8, 9, 10, 11, 12, e 13 considerando o impacto da variação dos tipos de janelas de tempo. Para cada instância são apresentados os valores do limite inferior, limite superior, Gap calculado como $Gap = \frac{LS-LI}{LS} \times 100\%$ e o tempo computacional. Foram realizados experimentos com os três tipos de janelas: JR, JM e JS.

Os resultados para as instâncias $(N = 15, T = 3)$ para o $CWTW$ e $SATW$ são mostrados nas Tabelas 8 e 9, respectivamente. Os resultados do $CWTW$ apontam que o LS de custo de solução do problema com janela restritiva (JR) é 16% em média maior que o custo considerando janela sem restrição (JS). Os resultados para o caso de janela mista (JM) apresentam uma diferença de 9% maior em média com relação a janela sem restrição. Ao compararmos o LS obtido pelo SATW, temos que em média o problema com JR é 14% mais custoso do que considerar JS, enquanto que JM é 5% mais custoso que JS. Se considerarmos uma comparação entre os dois métodos solucionando os diferentes tipos de janela, é observado que em alguns casos os dois algoritmos chegaram no mesmo custo de LS, porém, em média, o método $CWTW$ teve um custo e um tempo computacional médio menor do que o SATW. Sendo que o método $CWTW$ encontra soluções de custo 1% mais baixo que seu concorrente, executando o processo 33% mais rápido.

Os resultados para N=25 presentes nas Tabelas 10 e 11 demonstram que a diferença entre o LS das janelas aumenta com relação aos números obtidos em N=15. Observando os resultados do método $CWTW$ em N=25, é possível identificar uma diferença de 34% comparando JR com JS e 14% ao analisar os resultados de JM com JS. No caso $SATW$, a JR em relação a JS tem uma diferença de 29%. Porém, ao se comparar JM com JS, observamos uma diferença de 12%. Ao se comparar os dois métodos é identificado que o método $CWTW$ ainda tem um melhor desempenho, pois o custo do LS do método $CWTW$ é 5% menor que no método $SATW$. Ainda, nessas instâncias, o tempo computacional obtido no método $CWTW$ é em média 23% menor do que o obtido para o $SATW$.

Por fim, para as instâncias N=50, pelas Tabelas 12 e 13, observamos que a diferença dos resultados para o LS são de 25% e 14%, ao se comparar JR com JS e JR com JM, respectivamente, na implementação do método $CWTW$. Ao analisar a solução desenvolvida pelo método $SATW$ temos que a comparação entre JR e JS tem uma diferença de apenas 15% em média, e no momento em que observamos JR com JM a diferença se torna de 8%.

Nessas instâncias, é possível observar uma inversão na comparação entre os dois métodos. Nos casos anteriores, o método $CWTW$ obteve um melhor desempenho ao encontrar o custo do LS, porém, no caso de N=50, o método SATW obteve um custo 1% e 6% menor para JM e JR, respectivamente. Em média, o método SATW apresentou custos de LS 2% melhores que seu concorrente.

Para todas as instâncias foi definido um limite de 5 horas de processamento, sendo que nesses casos o método somente foi encerrado perante o término da iteração, isto é, todos os dados obtidos são resultado apenas de iterações completas. Após esse período, a melhor solução atual foi coletada, juntamente com os valores de LS e LI. Todas as instâncias com N=50 atingiram esse limite máximo, executando, em média, 8 iterações antes de serem finalizadas.

Os resultados de tempo computacional de todos os experimentos demonstram que não houve muita variação, pois, para o $SATW$ em relação a $CWTW$, a variação em média foi de no máximo 23% para as instâncias N=15, 26% para as instâncias N=25 e 1% para as instâncias N=50.

Tabela 8 – Resultados das instâncias (N = 15, T = 3) com janela de tempo (CWTW).

	Limitante Inferior			**Limitante Superior**			**Gap (%)**			**Tempo de CPU (s)**		
Instância	(JS)	(JM)	(JR)	(JS)	(JM)	(JR)	(JS)	(JM)	(JR)	(JS)	(JM)	(JR)
B15-0-T-3	907	909	892	1125	1387	1403	19,38	34,46	36,42	443	510	309
B15-1-T-3	901	888	837	1220	1316	1412	26,15	32,52	40,72	379	166	124
B15-2-T-3	895	895	923	1315	1422	1505	31,94	37,06	38,67	141	132	176
B15-3-T-3	924	881	856	1265	1386	1588	26,96	36,44	46,10	246	192	164
B15-4-T-3	857	846	867	1232	1387	1417	30,44	39,01	38,81	383	243	150
B15-5-T-3	885	848	848	1189	1268	1290	25,57	33,12	34,26	368	204	159
B15-6-T-3	924	929	918	1126	1320	1424	17,94	29,62	35,53	350	147	303
B15-7-T-3	876	848	872	1215	1219	1330	27,90	30,43	34,44	251	152	171
B15-8-T-3	911	943	901	1241	1328	1466	26,59	28,99	38,54	261	427	141
B15-9-T-3	872	884	833	1192	1224	1318	26,85	27,78	36,80	266	227	136
MédiB	895	887	875	1212	1326	1415	25,97	32,94	38,03	309	240	183

Tabela 9 – Resultados das instâncias (N = 15, T = 3) com janela de tempo (SATW).

	Limitante Inferior			**Limitante Superior**			**Gap (%)**			**Tempo de CPU (s)**		
Instância	(JS)	(JM)	(JR)	(JS)	(JM)	(JR)	(JS)	(JM)	(JR)	(JS)	(JM)	(JR)
B15-0-T-3	907	909	892	1156	1211	1453	21,58	24,94	38,61	370	230	299
B15-1-T-3	901	888	951	1220	1316	1446	26,17	32,52	34,23	382	559	481
B15-2-T-3	885	900	1018	1339	1422	1571	33,95	36,71	35,20	131	141	143
B15-3-T-3	924	982	957	1378	1433	1555	32,96	31,47	38,46	210	382	657
B15-4-T-3	860	841	857	1262	1387	1417	31,85	39,37	39,52	603	171	132
B15-5-T-3	891	817	850	1114	1268	1336	19,97	35,57	36,38	163	130	417
B15-6-T-3	924	928	917	1296	1320	1384	28,69	29,7	33,74	518	150	184
B15-7-T-3	876	845	873	1215	1274	1362	27,94	33,67	35,90	260	143	448
B15-8-T-3	910	943	899	1303	1329	1466	30,2	29,04	38,68	490	513	147
B15-9-T-3	870	884	848	1236	1258	1318	29,6	29,73	35,66	464	480	386
Média	895	894	906	1252	1322	1431	28,29	32,27	36,64	359	290	329

Tabela 10 – Resultados das instâncias (N = 25, T = 3) com janela de tempo (CWTW).

	Limitante Inferior			**Limitante Superior**			**Gap (%)**			**Tempo de CPU (s)**		
Instância	(JS)	(JM)	(JR)	(JS)	(JM)	(JR)	(JS)	(JM)	(JR)	(JS)	(JM)	(JR)
B25-0-T-3	923	848	753	1183	1222	1461	21,98	30,61	48,46	5994	1721	1595
B25-1-T-3	864	843	725	1151	1411	1503	24,93	40,26	51,76	3538	1739	1489
B25-2-T-3	1015	1079	712	1568	1720	1998	35,27	37,27	64,36	9237	2700	1504
B25-3-T-3	956	993	849	1344	1429	1578	28,87	30,51	46,20	5576	19592	1532
B25-4-T-3	970	1027	1022	1586	1755	1954	38,84	41,48	47,70	6025	12260	6433
B25-5-T-3	934	1052	839	1385	1960	2105	32,56	46,33	60,14	1455	4677	1591
B25-6-T-3	1009	985	1178	1560	1821	2670	35,32	45,91	55,88	11764	6174	2811
B25-7-T-3	977	948	1219	1815	2063	2406	46,17	54,05	49,33	8537	6301	2500
B25-8-T-3	879	944	793	1484	1609	1985	40,77	41,33	60,05	3443	6764	14867
B25-9-T-3	852	896	951	1219	1344	1617	30,11	33,33	41,19	11489	1573	2437
Média	938	962	904	1430	1633	1928	33,48	40,11	52,51	6706	6350	3676

8.3.3 Discussão

Os experimentos de N=15 e de N=25 foram os que apresentaram maiores semelhanças entre si, de modo que apresentaram como melhor solução o método *CWTW* enquanto que para os experimentos de N=50, o melhor método foi o *SATW*.

Tendo em vista os três tipos de janelas estudadas - sem restrição, mista e restritiva - já era esperado que o custo do limite superior (LS) das soluções com janelas sem restrição

Tabela 11 – Resultados das instâncias (N = 25, T = 3) com janela de tempo (SATW).

	Limitante Inferior			**Limitante Superior**			**Gap (%)**			**Tempo de CPU (s)**		
Instância	(JS)	(JM)	(JR)	(JS)	(JM)	(JR)	(JS)	(JM)	(JR)	(JS)	(JM)	(JR)
B25-0-T-3	916	948	993	1504	1519	1546	39,1	37,59	35,77	12781	1801	20126
B25-1-T-3	776	843	825	1448	1557	1563	46,39	45,86	47,22	1579	1656	1601
B25-2-T-3	1001	1087	1012	1833	2001	2181	45,39	45,68	53,60	10020	36185	1585
B25-3-T-3	956	769	1049	1392	1529	1678	31,36	49,71	37,49	3570	1596	1603
B25-4-T-3	961	1021	1021	1836	2258	2308	47,66	54,78	55,76	6654	2341	11024
B25-5-T-3	834	1051	1039	1533	1966	2205	45,61	46,54	52,88	1577	4507	1606
B25-6-T-3	987	973	1177	1584	1630	2377	37,67	40,31	50,48	20355	2159	2589
B25-7-T-3	965	942	1220	1517	1767	2635	36,36	46,69	53,70	10380	2456	2964
B25-8-T-3	885	948	1094	1534	1742	1874	42,31	45,58	41,62	10080	7747	19541
B25-9-T-3	803	696	953	1212	1302	1617	33,76	46,54	41,06	1642	1660	3438
Média	908	928	1038	1539	1727	1998	40,56	45,93	46,96	7864	6211	6608

Tabela 12 – Resultados das instâncias (N = 50, T = 3) com janela de tempo (CWTW).

	Limitante Inferior			**Limitante Superior**			**Gap (%)**			**Tempo de CPU (s)**		
Instância	(JS)	(JM)	(JR)	(JS)	(JM)	(JR)	(JS)	(JM)	(JR)	(JS)	(JM)	(JR)
B50-0-T-3	338	296	310	3088	3477	3799	89,05	91,49	91,84	18106	20047	19743
B50-1-T-3	335	354	355	3477	3639	3858	90,37	90,27	90,80	18298	19450	18936
B50-2-T-3	386	359	352	2817	3353	3722	86,30	89,29	90,54	18673	19173	18333
B50-3-T-3	333	328	315	2563	2857	3695	87,01	88,52	91,47	18772	19059	18535
B50-4-T-3	340	358	356	2639	2915	3733	87,12	87,72	90,46	19293	18591	18780
B50-5-T-3	346	403	352	4105	4362	5021	91,57	90,76	92,99	18004	19480	18167
B50-6-T-3	325	301	353	3565	3996	4400	90,88	92,47	91,98	19551	20251	19036
B50-7-T-3	346	343	313	2840	2970	3368	87,82	88,45	90,71	20392	19818	18425
B50-8-T-3	408	394	486	3266	3529	3922	87,51	88,84	87,61	19108	18529	19018
B50-9-T-3	393	381	404	3243	3567	3961	87,88	89,32	89,80	18013	18859	18546
Média	355	352	360	3160	3467	3948	88,55	89,71	90,82	18821	19326	18752

Tabela 13 – Resultados das instâncias (N = 50, T = 3) com janela de tempo (SATW).

	Limitante Inferior			**Limitante Superior**			**Gap (%)**			**Tempo de CPU (s)**		
Instância	(JS)	(JM)	(JR)	(JS)	(JM)	(JR)	(JS)	(JM)	(JR)	(JS)	(JM)	(JR)
B50-0-T-3	338	296	310	2989	3115	3229	88,70	90,50	90,40	20462	18418	20138
B50-1-T-3	335	354	355	3215	3384	3961	89,59	89,54	91,04	18593	19875	18424
B50-2-T-3	386	359	352	2425	3072	3422	84,08	88,31	89,71	19540	20013	18643
B50-3-T-3	333	328	315	2538	2765	3095	86,86	88,14	89,82	19847	18417	19449
B50-4-T-3	340	358	356	2956	3013	3589	88,48	88,12	90,08	18506	18881	19344
B50-5-T-3	346	403	652	4613	4662	4937	92,50	91,36	86,79	19469	18661	18821
B50-6-T-3	325	301	653	3275	3317	3549	90,06	90,93	81,60	19435	19219	20569
B50-7-T-3	346	343	313	3373	3444	3529	89,74	90,04	91,13	18452	18525	18892
B50-8-T-3	408	394	490	3848	3995	4070	89,39	90,14	87,96	18351	18440	20022
B50-9-T-3	393	381	404	3187	3472	3919	87,66	89,03	89,69	18089	19275	18140
Média	355	352	420	3242	3424	3730	88,71	89,61	88,82	19074	18972	19244

fosse menor do que com as janelas mistas e restritivas. Isso pode ser observado nos resultados obtidos em todos os experimentos realizados.

Para os experimentos com N=15, tendo em vista o limite superior da solução final, os dois métodos não apresentam grandes diferenças entre si, porém o destaque vai para o método *CWTW* no quesito de tempo de processamento. Sendo esse método 33% mais rápido que seu concorrente.

No caso de N=25, a comparação entre os métodos *CWTW* e SATW continuou próxima no quesito do LS. Entretanto, a diferença do tempo computacional entre as duas

soluções diminuiu, sendo agora de apenas 23%. Assim, a mesma tendência dos resultados das variantes propostas para o algoritmo de Rahim (apresentados na Seção 8.2.3) pode ser observada para o caso do SIRPTW, i.e., a diferença entre os tempos computacionais diminuí enquanto a quantidade de clientes das instâncias aumenta.

Por fim, para as instâncias N=50, houve uma inversão da melhor solução em relação aos casos anteriores. Ao analisar os resultados, o limite superior apresentado pelo $SATW$ obteve um melhor resultado que $CWTW$. Entretanto, devido a restrição de 5 horas de processamento, não foi possível realizar a comparação entre os tempos de processamento. Porém, se levarmos em conta a identificação da mesma tendência observada nos resultados da Seção 8.2.3, é presumível que o resultado final encontrado pelo $SATW$ sem a restrição de 5 horas, isto é, executando o processamento até a convergência do limite inferior e superior, apresente um melhor resultado do que o encontrado pelo $CWTW$ nas mesmas condições, em relação ao custo da solução e tempo computacional.

Dessa forma, os experimentos mostram que o método $SATW$ é o mais adequado para as instâncias maiores e mais complexas. E seu concorrente, o método $CWTW$, se torna a melhor alternativa para instâncias menores.

8.4 Avaliação do algoritmo para o SIRPTW em um problema real

A seguir são detalhados as instâncias, resultados e a discussão do experimento com uma instância que é gerada a partir de uma distribuição real e que inclui janelas de tempo.

8.4.1 Benchmark de instâncias

Foi realizado também um experimento considerando uma instância com dados reais, simulando a distribuição de uma empresa no período de 3 dias. Com o conhecimento obtido pela experiência profissional no mercado empresarial do autor, foram separados aleatoriamente 30 endereços na cidade de São Paulo–SP, sendo 29 como clientes e 1 para o depósito da operação. A dispersão dos clientes com relação ao depósito é representado na Figura 5, em que os círculos representam os clientes e o quadrado indica o depósito.

A demanda e a janela de tempo de cada cliente, detalhados na Tabela 14, foram definidos com o conhecimento prévio de casos reais em um ambiente de distribuição de

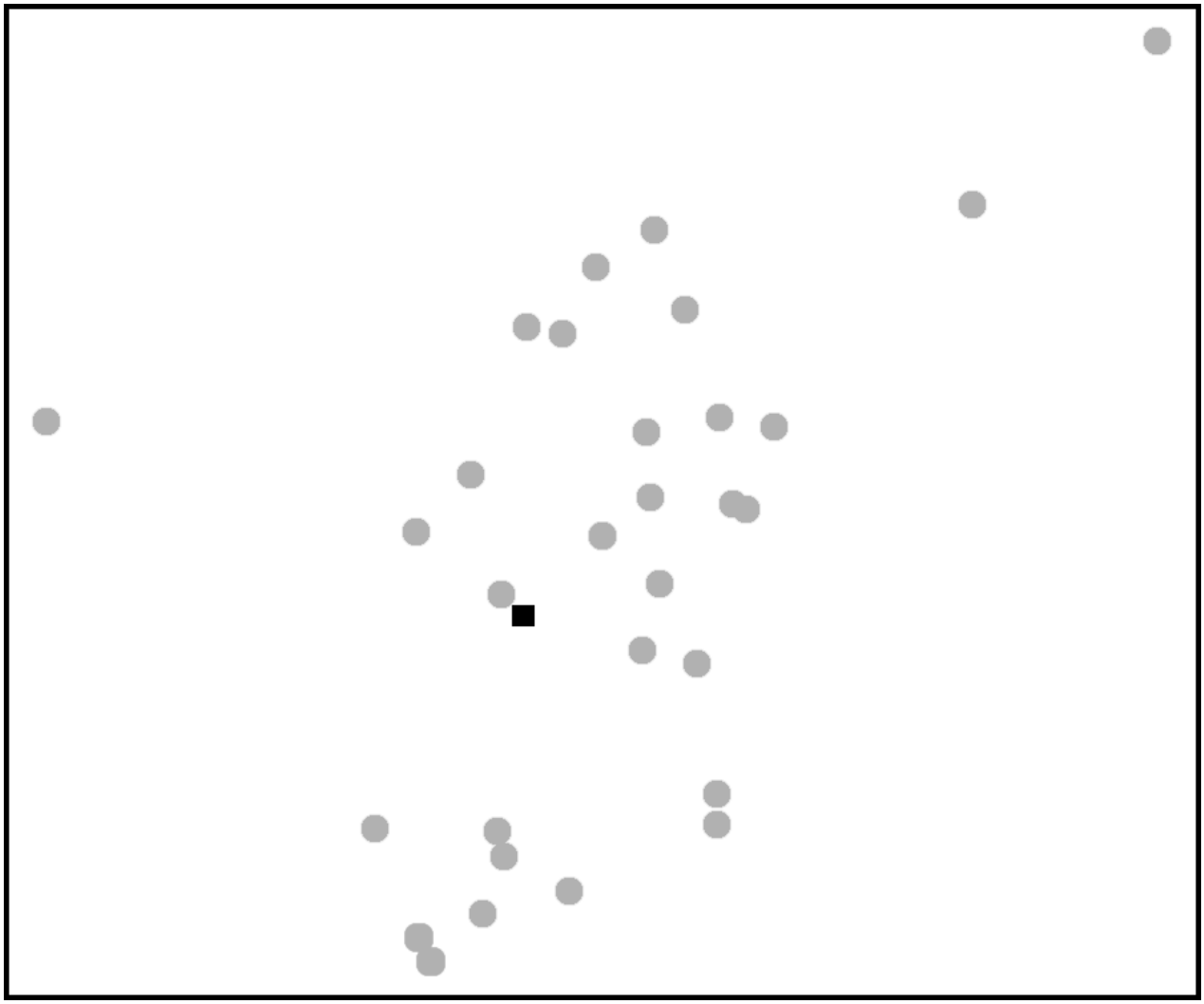

Figura 5 – Dispersão dos clientes no caso real

produtos na mesma região estabelecida pelos endereços. Os outros parâmetros estabelecidos nesse modelo foram mantidos pois eles condizem com características observadas em um cenário real de distribuição.

Com auxílio da Distance Matrix API,[1] com licença gratuita disponível pela Google, foi construída uma matriz de distância entre todos os endereços selecionados para simular o deslocamento em tempo real pelas vias da cidade. Dessa forma, uma instância foi gerada com características reais de uma distribuição da área metropolitana da cidade de São Paulo.

Em seguida, foram realizados experimentos com os dois métodos desenvolvidos nesse trabalho ($CWTW$ e $SATW$) e com uma adaptação do algoritmo guloso descrito

[1] https://developers.google.com/maps/documentation/distance-matrix/?hl=pt-br Acessado em: 02/09/2017

Tabela 14 – Dados de demanda e janela de atendimento no caso real

Cliente	Demanda	Hora Inicio	Hora Fim
1	7,28	15:25	16:30
2	22,60	15:30	15:55
3	2,43	15:45	16:20
4	18,41	00:00	23:59
5	10,14	10:40	12:00
6	4,32	00:00	23:59
7	5,86	10:00	11:50
8	16,46	13:25	13:50
9	31,35	00:00	23:59
10	5,26	14:35	15:20
11	12,25	00:00	23:59
12	1,33	15:00	16:15
13	9,56	15:20	15:55
14	5,08	00:00	23:59
15	16,92	10:40	11:35
16	0,03	00:00	23:59
17	2,34	14:45	16:20
18	1,66	00:00	23:59
19	8,97	15:30	16:00
20	4,17	06:30	06:45
21	6,14	00:00	23:59
22	16,15	06:30	07:55
23	6,36	00:00	23:59
24	5,29	00:00	23:59
25	3,19	00:00	23:59
26	3,63	06:30	08:10
27	0,77	00:00	23:59
28	3,30	06:30	07:50
29	9,82	00:00	23:59
30	1,53	06:30	07:30

em (CAMPBELL; SAVELSBERGH, 2004) que inclui restrições de janela de tempo. Esse algoritmo guloso seleciona clientes que estarão mais próximos de ficar sem estoque no período. A quantidade de entrega para esses clientes é igual a capacidade total de inventário menos a quantidade atual do produto, isto é, o estoque do cliente é reabastecido até sua totalidade. Em seguida, se sobrou alguma capacidade restante no veículo, os produtos serão então distribuídos entre clientes que irão ficar sem estoque no período seguinte. A sequência de entrega é calculada usando uma heurística de inserção considerando a restrição de janela de tempo.

Como nesse método guloso as entregas devem ser realizadas no dia em que foram agendadas. Caso o método de inserção não conseguisse montar a sequencia devido a restrição de janela, a entrega em questão é atribuída a uma nova rota.

A comparação com o algoritmo guloso é valida pois é uma estratégia comumente usada por empresas que se deparam com esse problema, que devido a falta de uma solução eficaz, acabam adotando em sua operação métodos como esse.

8.4.2 Resultados

Na Tabela 15 são comparados os valores do limite inferior, limite superior, Gap e o tempo computacional da execução dos métodos $CWTW$ e o $SATW$ na instância real.

O método $CWTW$ conseguiu encontrar uma solução com limite superior 12% menor que o método $SATW$. Ao observarmos o Gap das soluções, também é possível identificar que ambos são razoavelmente aceitáveis dada a complexidade da instância gerada. Entretanto, com relação ao tempo computacional, o método $CWTW$ teve um desempenho pior que seu concorrente. De modo que o método $SATW$ foi 42% mais rápido para encontrar a solução final que $CWTW$.

Tabela 15 – Resultado da instância do problema real com janela de tempo.

Instância	**Limitante Inferior** (CWTW)	(SATW)	**Limitante Superior** (CWTW)	(SATW)	**Gap (%)** (CWTW)	(SATW)	**Tempo de CPU (s)** (CWTW)	(SATW)
R30-0-T-3	2152	2347	3019	3454	28,72	32,04	14012	8092

O custo da solução e tempo de CPU em segundos do algoritmo guloso na instância real são mostrados na Tabela 16. O tempo computacional é baixo, mas não compensa o custo elevado da solução encontrada.

Tabela 16 – Resultado do algoritmo guloso no problema real.

Instância	**Custo da solução** (Guloso)	**Tempo de CPU (s)** (Guloso)
RG30-0-T-3	4018	249

8.4.3 Discussão

Ao considerarmos os resultados da execução dos algoritmos propostos para o problema SIRPTW com dados encontrados em uma operação de distribuição real em uma

grande cidade, obtemos resultados satisfatórios, pois, estes não estão fora do esperado ao compararmos com os outros experimentos realizados com instâncias geradas aleatoriamente.

Em geral, os resultados do método $CWTW$ foram melhores que de seu concorrente $SATW$. No quesito custo da solução final, ou LS, o método $CWTW$ encontrou uma solução 12% menos custosa que o outro método. Se formos colocar o peso de uma operação real de distribuição, essa diferença deve ser considerado na escolha do método que será utilizado. Outra diferença entre os resultados está no tempo computacional. O método $CWTW$ teve um desempenho 42% mais lento que seu concorrente. Porém, devido a complexidade dessa operação, não acreditamos que esse fator seja mais relevante do a diferença de 12% no custo final da solução. Dessa forma, o método $CWTW$ se mostrou o método mais adequado para solucionar o SIRPTW nessa operação de distribuição real, tendo em visa que, pode haver uma inversão do melhor método conforme a quantidade de clientes da instância aumente, como discutido na Seção 8.3.3.

Ao comparar o resultado obtido com o algoritmo guloso, é observado que o algoritmo $CWTW$ encontra uma solução 25% menos custosa.

9 Conclusões e trabalhos futuros

Neste trabalho foi investigado o problema de roteamento de veículo com demanda estocástica (SIRP) juntamente com o problema que inclui janelas de tempo (SIRPTW). Esses problemas são constantemente encontrados em diversos ramos da industria e distribuição de produtos. A sua complexidade computacional é um dos principais fatores para que a maioria das soluções apresentadas até o momento estejam baseadas em heurísticas e politicas para atender o portfólio de clientes. Para resolver o SIRPTW, foi primeiramente proposto um modelo, que teve como base o trabalho de Rahim et al. (2014), e que atende as principais características que as empresas enfrentam.

9.1 Resumo das contribuições

Dada a motivação inicial para resolver o problema de roteamento e estoque com demanda estocástica e janela de tempo, esse trabalho resultou em diversas contribuições originais para as áreas de Inteligência Artificial, Otimização Combinatória e Inteligência de Sistemas. Além disso, considerando o custo total da cadeia de suprimento, o modelo proposto será uma ferramenta útil na operação dos sistemas de produção e distribuição. As principais contribuições desse trabalho são:

- No Capitulo 6 foram implementadas três variações para solucionar o problema SIRP. Esses algoritmos foram adaptados para substituir o Algoritmo 2, que tem como objetivo buscar soluções viáveis para o problema de roteamento de veículo em cada período do planejamento. Os métodos implementados têm como base as técnicas de agrupamento por Centróide, Simulated Annealing e Monte Carlo Savings.
- No Capitulo 7 é apresentado um modelo que aborda o problema de estoque e roteirização com demanda estocástica e janela de tempo (SIRPTW). Tal modelo foi elaborado visando a aproximação do problema para casos mais realistas. Com isso, além da janela de tempo, foram adicionadas ao modelo as Restrições (56) e (57), que tratam o nível de confiança individualizado para cada cliente, juntamente com a Restrição (48), que restringe a capacidade máxima de estoque de cada cliente.

- Além do modelo apresentado, na Seção 7.5 foram propostos dois algoritmos que buscam solucionar o SIRPTW, denominados de $CWTW$ e $SATW$, os quais têm como base os algoritmos Clarke & Wright e Simulated Annealing, respectivamente.
- Foi desenvolvido também, na Seção 8.1, um gerador de instâncias de SIRP e SIRPTW, o qual permitirá que algoritmos nesta área sejam comparados.
- No Capitulo 8 são mostrados os resultados dos experimentos tanto com instâncias sintéticas, as quais são geradas aleatoriamente, quanto com uma instância real, baseada na experiência profissional no mercado empresarial e em cenários reais de distribuição na cidade de São Paulo.

Dentre os algoritmos implementados para o SIRP, todos apresentaram vantagens e desvantagens. Os experimentos mostraram que o algoritmo Rahim com Simulated Annealing é uma boa alternativa quando desejamos resolver instâncias grandes e complexas. Para instâncias menores, o melhor continua sendo o algoritmo original, i.e., o algoritmo de Rahim com C&W. A redução do custo para instâncias com 50 clientes do algoritmo Rahim com Simulated Annealing em comparação ao algoritmo de Rahim com C&W chega a 17% em média. Essa redução se torna muito significativa quando analisamos a relevância do problema SIRP na operação logística.

Os resultados para SIRPTW mostram que o custo total da operação e o intervalo de tempo das entregas têm impactos diretos nas variáveis de decisão . Os resultados obtidos para SIRPTW em instâncias sintéticas, mostraram que o método baseado em Simulated Annealing se torna cada vez melhor quanto maior for a instância em que os testes são aplicados. Sendo que nas instâncias com 50 clientes, esse algoritmo obteve um resultado melhor do que seu concorrente, o $CWTW$. Porém, ambos algoritmos para N=50 não conseguem convergir em 5 horas de processamento.

Após os testes com instâncias sintéticas, os métodos $CWTW$ e $SATW$ foram avaliados com uma instância real. Os dados dessa instância são baseados na demanda real de uma distribuição, o que é extremamente significativo considerando que um dos objetivos principais desse trabalho é aproximar o problema de casos encontrados na realidade no ramo de distribuição. Os resultados desse teste mostram que o método $CWTW$ é a melhor opção para essa instância de distribuição real quando comparado com o algorimo guloso e o algoritmo $SATW$. O custo de operação encontrado pelo algoritmo $CWTW$ foi 12% menos custosa que o método SATW. Mesmo considerando que o método $CWTW$ foi 42%

mais lento para a instância real, esse fator não é tão impactante quanto a economia de 12% em uma operação de distribuição.

9.2 Publicações geradas e em andamento

Os algoritmos desenvolvidos como variantes para o algoritmo de Rahim foram publicados no XIII Simpósio Brasileiro de Sistemas de Informação - SBSI 2017, um dos principais simpósios na área de Sistemas de Informação.

Estão sendo escritos dois artigos que serão submetidos para o XIV Simpósio Brasileiro de Sistemas de Informação - SBSI 2018. O primeiro contém a revisão sistemática sobre IRPTW e o segundo tem como objetivo apresentar o modelo proposto para SIRPTW e os resultados obtidos nos experimentos do mesmo.

9.3 Trabalhos futuros

Entre as possíveis extensões deste trabalho estão a aplicação de:

- **Outros métodos de relaxação:** Como introduzido no Capitulo 2 existem outros métodos de relaxação que buscam simplificar o problema original para encontrar a solução ótima ou quase ótima para um modelo de programação inteira. Os principais métodos são a relaxação Surrogate, a relaxação Lagrange/Surrogate (*lagsur*), a relaxação combinada Lagrangeana-Surrogate e a relaxação de programação linear.
- **Método de geração de colunas:** Por se tratar de um modelo de programação linear, o modelo proposto pode ser resolvido por outros métodos além de relaxações. Uma das formas de resolver um problema de programação linear é por geração de colunas. Este método, inicialmente proposto em (JR; FULKERSON, 1958) e detalhado em (DESROSIERS; LÜBBECKE, 2005), visa simplificar um problema para encontrar a solução de forma mais simples, assim como os métodos de relaxação. Esse método é indicado para resolver modelos de programação inteira que contém um grande numero de variáveis, como o modelo proposto no Capitulo 7. O método de geração de colunas tem como vantagem que nem todas as possibilidades precisam ser enumeradas durante a solução do modelo. Para tal, o problema é primeiro formulado como um problema mestre restrito. Este problema tem o menor número

possível de variáveis e novas variáveis são trazidas para o modelo conforme necessário. Teoricamente isso é possível pois a maioria das variáveis não farão parte da solução ótima e assumirão o valor zero na solução ideal, ou seja, apenas um subconjunto de variáveis deve ser considerado ao resolver o problema.

Referências[1]

ADELMAN, D. A price-directed approach to stochastic inventory/routing. *Operations Research*, INFORMS, v. 52, n. 4, p. 499–514, 2004. Citado 4 vezes nas páginas 21, 37, 42 e 84.

AGHEZZAF, E.-H. Robust distribution planning for supplier-managed inventory agreements when demand rates and travel times are stationary. *Journal of the Operational Research Society*, Nature Publishing Group, v. 59, n. 8, p. 1055–1065, 2008. Citado 2 vezes nas páginas 37 e 42.

ANILY, S.; FEDERGRUEN, A. One warehouse multiple retailer systems with vehicle routing costs. *Management Science*, INFORMS, v. 36, n. 1, p. 92–114, 1990. Citado na página 36.

ANILY, S.; FEDERGRUEN, A. Two-echelon distribution systems with vehicle routing costs and central inventories. *Operations Research*, INFORMS, v. 41, n. 1, p. 37–47, 1993. Citado na página 36.

ARCHETTI, C.; DOERNER, K.; TRICOIRE, F. A heuristic algorithm for the free newspaper delivery problem. *European Journal of Operational Research*, v. 230, n. 2, p. 245–257, 2013. Citado 3 vezes nas páginas 44, 45 e 47.

BAKER, E. K.; SCHAFFER, J. R. Solution improvement heuristics for the vehicle routing and scheduling problem with time window constraints. *American Journal of Mathematical and Management Sciences*, Taylor & Francis, v. 6, n. 3-4, p. 261–300, 1986. Citado na página 32.

BARD, J. F. et al. A decomposition approach to the inventory routing problem with satellite facilities. *Transportation science*, INFORMS, v. 32, n. 2, p. 189–203, 1998. Citado 2 vezes nas páginas 37 e 38.

BARNES-SCHUSTER, D.; BASSOK, Y. Direct shipping and the dynamic single-depot/multi-retailer inventory system. *European Journal of Operational Research*, Elsevier, v. 101, n. 3, p. 509–518, 1997. Citado 2 vezes nas páginas 37 e 38.

BELFIORE, P. P.; COSTA, O. L. d. V.; FÁVERO, L. P. L. Problema de estoque e roteirização: revisão bibliográfica. *Production*, scielo, v. 16, p. 442 – 454, 12 2006. Citado 2 vezes nas páginas 20 e 22.

BENOIST, T. et al. High-performance local search for solving real-life inventory routing problems. In: SPRINGER. *International Workshop on Engineering Stochastic Local Search Algorithms*. [S.l.], 2009. p. 105–109. Citado na página 22.

BERMAN, O.; LARSON, R. C. Deliveries in an inventory/routing problem using stochastic dynamic programming. *Transportation Science*, INFORMS, v. 35, n. 2, p. 192–213, 2001. Citado 2 vezes nas páginas 37 e 39.

BERTAZZI, L. et al. A stochastic inventory routing problem with stock-out. *Transportation Research Part C: Emerging Technologies*, Elsevier, v. 27, p. 89–107, 2013. Citado 3 vezes nas páginas 37, 43 e 84.

[1] De acordo com a Associação Brasileira de Normas Técnicas. NBR 6023.

BERTAZZI, L.; PALETTA, G.; SPERANZA, M. G. Deterministic order-up-to level policies in an inventory routing problem. *Transportation Science*, INFORMS, v. 36, n. 1, p. 119–132, 2002. Citado na página 36.

BERTAZZI, L.; SAVELSBERGH, M.; SPERANZA, M. G. Inventory routing. In: GOLDEN, B.; RAGHAVAN, S.; WASIL, E. (Ed.). *The Vehicle Routing Problem: Latest Advances and New Challenges*. Boston, MA: Springer US, 2008. p. 49–72. Citado na página 35.

BINGLEI, X.; SHI, A.; JIAN, W. Stochastic inventory routing problem under b2c e-commerce. In: IEEE. *e-Business Engineering, 2005. ICEBE 2005. IEEE International Conference on*. [S.l.], 2005. p. 630–633. Citado 3 vezes nas páginas 37, 39 e 84.

BITRAN, G. R.; YANASSE, H. Computational complexity of the capacitated lot size problem. *Management Science*, v. 28, n. 10, p. 1271–81, 1982. Citado na página 58.

BRAMEL, J.; SIMCHI-LEVI, D. A location based heuristic for general routing problems. *Operations research*, INFORMS, v. 43, n. 4, p. 649–660, 1995. Citado na página 36.

CÁCERES-CRUZ, J. et al. Combining monte carlo simulation with heuristics for solving the inventory routing problem with stochastic demands. p. 274, 2012. Citado 3 vezes nas páginas 37, 41 e 84.

CAMPBELL, A. et al. The inventory routing problem. In: *Fleet management and logistics*. [S.l.]: Springer, 1998. p. 95–113. Citado 3 vezes nas páginas 35, 37 e 41.

CAMPBELL, A.; CLARKE, L.; SAVELSBERGH, M. Inventory routing in practice. In: TOTH, P.; VIGO, D. (Ed.). *The Vehicle Routing Problem*. [S.l.]: Society for Industrial and Applied Mathematics, 2002, (Monographs on Discrete Mathematics and Applications). p. 309–330. Citado na página 35.

CAMPBELL, A.; SAVELSBERG, M. Delivery volume optimization. *Transportation Science*, v. 38, n. 2, p. 210–223, 2004. Citado 2 vezes nas páginas 44 e 45.

CAMPBELL, A. M.; SAVELSBERGH, M. W. A decomposition approach for the inventory-routing problem. *Transportation science*, INFORMS, v. 38, n. 4, p. 488–502, 2004. Citado 2 vezes nas páginas 36 e 96.

CHAN, L. M. A.; FEDERGRUEN, A.; SIMCHI-LEVI, D. Probabilistic analyses and practical algorithms for inventory-routing models. *Operations Research*, INFORMS, v. 46, n. 1, p. 96–106, 1998. Citado na página 36.

CHIANG, W.-C.; RUSSELL, R. A. Simulated annealing metaheuristics for the vehicle routing problem with time windows. *Annals of Operations Research*, Springer, v. 63, n. 1, p. 3–27, 1996. Citado na página 32.

CHIEN, T. W.; BALAKRISHNAN, A.; WONG, R. T. An integrated inventory allocation and vehicle routing problem. *Transportation Science*, INFORMS, v. 23, n. 2, p. 67–76, 1989. Citado na página 35.

CLARKE, G.; WRIGHT, J. W. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, INFORMS, v. 12, n. 4, p. 568–581, 1964. Citado 3 vezes nas páginas 29, 38 e 62.

COELHO, L. C.; CORDEAU, J.-F.; LAPORTE, G. Thirty years of inventory routing. *Transportation Science*, INFORMS, v. 48, n. 1, p. 1–19, 2013. Citado 8 vezes nas páginas 20, 21, 23, 33, 34, 35, 36 e 37.

CORDEAU, J.-F. et al. *The VRP with time windows*. [S.l.]: Montréal: Groupe d'études et de recherche en analyse des décisions, 2000. Citado na página 43.

COUSINEAU-OUIMET, K. A tabu search heuristic for the inventory routing problem. In: *Proceedings of 37th Annual ORSNZ Conference*. [S.l.: s.n.], 2002. Citado na página 36.

CROES, G. A. A method for solving traveling-salesman problems. *Operations research*, INFORMS, v. 6, n. 6, p. 791–812, 1958. Citado na página 46.

CUSTÓDIO, A. L.; OLIVEIRA, R. C. Redesigning distribution operations: a case study on integrating inventory management and vehicle routes design. *International Journal of Logistics: Research and Applications*, Taylor & Francis, v. 9, n. 2, p. 169–187, 2006. Citado 2 vezes nas páginas 37 e 39.

DANTZIG, G. B.; RAMSER, J. H. The truck dispatching problem. *Management science*, Informs, v. 6, n. 1, p. 80–91, 1959. Citado na página 29.

DESROSIERS, J.; LÜBBECKE, M. E. A primer in column generation. In: *Column generation*. [S.l.]: Springer, 2005. p. 1–32. Citado na página 101.

EL-SHERBENY, N. A. Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods. *Journal of King Saud University-Science*, Elsevier, v. 22, n. 3, p. 123–131, 2010. Citado na página 32.

ESPEJO, L. G. A.; GALVÃO, R. D. O uso das relaxações Lagrangeana e Surrogate em problemas de programação inteira. *Pesquisa Operacional*, scielo, v. 22, p. 387 – 402, 07 2002. Citado na página 26.

FEDERGRUEN, A.; PRASTACOS, G.; ZIPKIN, P. H. An allocation and distribution model for perishable products. *Operations Research*, INFORMS, v. 34, n. 1, p. 75–82, 1986. Citado 2 vezes nas páginas 37 e 38.

FEDERGRUEN, A.; ZIPKIN, P. A combined vehicle routing and inventory allocation problem. *Operations Research*, INFORMS, v. 32, n. 5, p. 1019–1037, 1984. Citado 4 vezes nas páginas 21, 22, 36 e 37.

FEDERGRUEN, A.; ZIPKIN, P. An efficient algorithm for computing optimal (s, s) policies. *Operations research*, INFORMS, v. 32, n. 6, p. 1268–1285, 1984. Citado 2 vezes nas páginas 37 e 38.

FISHER, M. L. The Lagrangian relaxation method for solving integer programming problems. *Management science*, INFORMS, v. 27, n. 1, p. 1–18, 1981. Citado 3 vezes nas páginas 27, 28 e 60.

FISHER, M. L.; JAIKUMAR, R. A generalized assignment heuristic for vehicle routing. *Networks*, Wiley Online Library, v. 11, n. 2, p. 109–124, 1981. Citado na página 36.

FISHER, M. L.; JÖRNSTEN, K. O.; MADSEN, O. B. Vehicle routing with time windows: Two optimization algorithms. *Operations research*, INFORMS, v. 45, n. 3, p. 488–492, 1997. Citado na página 32.

GAUR, V.; FISHER, M. L. A periodic inventory routing problem at a supermarket chain. *Operations Research*, INFORMS, v. 52, n. 6, p. 813–822, 2004. Citado 2 vezes nas páginas 37 e 39.

GEIGER, M. J.; SEVAUX, M. Practical inventory routing: A problem definition and an optimization method. *arXiv preprint arXiv:1102.5635*, 2011. Citado 2 vezes nas páginas 37 e 40.

GENDREAU, M.; HERTZ, A.; LAPORTE, G. New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research*, INFORMS, v. 40, n. 6, p. 1086–1094, 1992. Citado na página 51.

GOLDEN, B.; ASSAD, A.; DAHL, R. Analysis of a large scale vehicle routing problem with an inventory component. *Large scale systems*, North-Holland, v. 7, n. 2-3, p. 181–190, 1984. Citado 2 vezes nas páginas 37 e 38.

HARMANANI, H. M. et al. A simulated annealing algorithm for the capacitated vehicle routing problem. In: *Proceedings of the ISCA 26th International Conference on Computers and Their Applications*. [S.l.: s.n.], 2011. p. 96–101. Citado 2 vezes nas páginas 30 e 66.

HELD, M.; KARP, R. M. The traveling-salesman problem and minimum spanning trees: Part ii. *Mathematical programming*, Springer, v. 1, n. 1, p. 6–25, 1971. Citado na página 28.

HUANG, S.-H.; LIN, P.-C. A modified ant colony optimization algorithm for multi-item inventory routing problems with demand uncertainty. *Transportation Research Part E: Logistics and Transportation Review*, Elsevier, v. 46, n. 5, p. 598–611, 2010. Citado 3 vezes nas páginas 21, 37 e 40.

HVATTUM, L. M.; LØKKETANGEN, A. Using scenario trees and progressive hedging for stochastic inventory routing problems. *Journal of Heuristics*, Springer, v. 15, n. 6, p. 527–557, 2009. Citado 2 vezes nas páginas 37 e 40.

HVATTUM, L. M.; LØKKETANGEN, A.; LAPORTE, G. Scenario tree-based heuristics for stochastic inventory-routing problems. *INFORMS Journal on Computing*, INFORMS, v. 21, n. 2, p. 268–285, 2009. Citado 3 vezes nas páginas 21, 37 e 40.

IASSINOVSKAIA, G.; LIMBOURG, S.; RIANE, F. The inventory-routing problem of returnable transport items with time windows and simultaneous pickup and delivery in closed-loop supply chains. *International Journal of Production Economics*, v. 183, p. 570–582, 2017. Citado 3 vezes nas páginas 44, 45 e 48.

JAILLET, P. et al. Delivery cost approximations for inventory routing problems in a rolling horizon framework. *Transportation Science*, INFORMS, v. 36, n. 3, p. 292–300, 2002. Citado 2 vezes nas páginas 37 e 39.

JARUGUMILLI, S.; GRASMAN, S. E. Rfid-enabled inventory routing problems. *International Journal of Manufacturing Technology and Management*, Inderscience Publishers, v. 10, n. 1, p. 92–105, 2006. Citado 2 vezes nas páginas 37 e 40.

JARUGUMILLI, S.; GRASMAN, S. E.; RAMAKRISHNAN, S. A simulation framework for real-time management and control of inventory routing decisions. In: IEEE. *Proceedings of the 2006 Winter simulation Conference*. [S.l.], 2006. p. 1485–1492. Citado 2 vezes nas páginas 37 e 40.

JIA, T. et al. Integrated inventory routing problem with quality time windows and loading cost for deteriorating items under discrete time. *Mathematical Problems in Engineering*, vol.2014, 2014. Citado 3 vezes nas páginas 44, 45 e 47.

JR, L. R. F.; FULKERSON, D. R. A suggested computation for maximal multi-commodity network flows. *Management Science*, INFORMS, v. 5, n. 1, p. 97–101, 1958. Citado na página 101.

JUAN, A. A. et al. A simheuristic algorithm for the single-period stochastic inventory-routing problem with stock-outs. *Simulation Modelling Practice and Theory*, Elsevier, v. 46, p. 40–52, 2014. Citado 4 vezes nas páginas 21, 37, 41 e 84.

KLEYWEGT, A. J.; NORI, V. S.; SAVELSBERGH, M. W. The stochastic inventory routing problem with direct deliveries. *Transportation Science*, INFORMS, v. 36, n. 1, p. 94–118, 2002. Citado 3 vezes nas páginas 21, 37 e 42.

KLEYWEGT, A. J.; NORI, V. S.; SAVELSBERGH, M. W. Dynamic programming approximations for a stochastic inventory routing problem. *Transportation Science*, INFORMS, v. 38, n. 1, p. 42–70, 2004. Citado 2 vezes nas páginas 37 e 42.

KOHL, N.; MADSEN, O. B. An optimization algorithm for the vehicle routing problem with time windows based on lagrangian relaxation. *Operations research*, INFORMS, v. 45, n. 3, p. 395–406, 1997. Citado na página 32.

KVANLI, A.; PAVUR, R.; KEELING, K. *Concise managerial statistics*. [S.l.]: Cengage Learning, 2005. Citado 2 vezes nas páginas 57 e 76.

LI, K. et al. An inventory–routing problem with the objective of travel time minimization. *European Journal of Operational Research*, v. 236, n. 3, p. 936–945, ago. 2014. Citado 4 vezes nas páginas 44, 45, 49 e 76.

LIN, S.; KERNIGHAN, B. W. An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, INFORMS, v. 21, n. 2, p. 498–516, 1973. Citado 2 vezes nas páginas 31 e 69.

LIU, S.-C.; LEE, W.-T. A heuristic method for the inventory routing problem with time windows. *Expert Systems with Applications*, v. 38, n. 10, p. 13223–13231, 2011. Citado 3 vezes nas páginas 44, 45 e 46.

LORENA, L. A. N.; PEREIRA, M. A.; SALOMÃO, S. N. A. A relaxação Lagrangeana/Surrogate e o método de geração de colunas: novos limitantes e novas colunas. *Pesquisa Operacional*, scielo, v. 23, p. 29 – 47, 01 2003. Citado na página 26.

LOU, S.; WU, Y.; XIAO, J. Study on integrated inventory-routing problems. In: IEEE. *Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on*. [S.l.], 2009. v. 1, p. 42–46. Citado 3 vezes nas páginas 37, 40 e 84.

MINKOFF, A. S. A Markov decision model and decomposition heuristic for dynamic vehicle dispatching. *Operations Research*, INFORMS, v. 41, n. 1, p. 77–90, 1993. Citado 2 vezes nas páginas 37 e 38.

NARCISO, M. G.; LORENA, L. A. N. Uma aplicação da relaxação Lagrangeana/Surrogate ao problema simétrico do caixeiro viajante usando um método de subgradientes melhorado. Citado na página 26.

NARCISO, M. G.; LORENA, L. A. N. Lagrangean/Surrogate relaxation for generalized assignment problems. *European Journal of Operational Research*, Elsevier, v. 114, n. 1, p. 165–177, 1999. Citado na página 26.

NOLZ, P. C.; ABSI, N.; FEILLET, D. A stochastic inventory routing problem for infectious medical waste collection. *Networks*, Wiley Online Library, v. 63, n. 1, p. 82–95, 2014. Citado 2 vezes nas páginas 37 e 41.

OLIVEIRA, R. A. de C.; DELGADO, K. V.; MOREIRA, D. A. Sistema para roteamento de veículos capacitados aplicando métodos de Monte Carlo. *iSys-Revista Brasileira de Sistemas de Informação*, v. 8, n. 3, p. 42–63, 2016. Citado 3 vezes nas páginas 31, 66 e 70.

PÉREZ, J. A. M.; MORENO-VEGA, J. M.; MARTIN, I. R. Variable neighborhood tabu search and its application to the median cycle problem. *European Journal of Operational Research*, Elsevier, v. 151, n. 2, p. 365–378, 2003. Citado na página 46.

QU, W. W.; BOOKBINDER, J. H.; IYOGUN, P. An integrated inventory–transportation system with modified periodic policy for multiple products. *European Journal of Operational Research*, Elsevier, v. 115, n. 2, p. 254–269, 1999. Citado 2 vezes nas páginas 37 e 38.

RAHIM, M. K. I. A. et al. Modelling and solving the multiperiod inventory-routing problem with stochastic stationary demand rates. *International Journal of Production Research*, Taylor & Francis, v. 52, n. 14, p. 4351–4363, 2014. Citado 24 vezes nas páginas 8, 9, 21, 22, 23, 24, 37, 43, 53, 54, 55, 56, 57, 61, 63, 64, 65, 73, 76, 77, 79, 84, 86 e 99.

REIMAN, M. I.; RUBIO, R.; WEIN, L. M. Heavy traffic analysis of the dynamic stochastic inventory-routing problem. *Transportation Science*, INFORMS, v. 33, n. 4, p. 361–380, 1999. Citado 2 vezes nas páginas 37 e 38.

ROCKAFELLAR, R. T.; WETS, R. J.-B. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of operations research*, INFORMS, v. 16, n. 1, p. 119–147, 1991. Citado na página 40.

RUSDIANSYAH, A.; TSAO, D.-B. An integrated model of the periodic delivery problems for vending-machine supply chains. *Journal of Food Engineering*, v. 70, n. 3, p. 421–434, 2005. Citado 3 vezes nas páginas 44, 45 e 46.

RUSSELL, R. A. Hybrid heuristics for the vehicle routing problem with time windows. *Transportation science*, INFORMS, v. 29, n. 2, p. 156–166, 1995. Citado na página 32.

SCHULZE, J.; FAHLE, T. A parallel algorithm for the vehicle routing problem with time window constraints. *Annals of Operations Research*, Springer, v. 86, p. 585–607, 1999. Citado na página 32.

SENNE, E. L.; LORENA, L. A. A lagrangean/surrogate approach to p-median problems. 1999. Citado na página 26.

SHAW, P. Using constraint programming and local search methods to solve vehicle routing problems. In: SPRINGER. *International Conference on Principles and Practice of Constraint Programming*. [S.l.], 1998. p. 417–431. Citado na página 47.

SHIN, K.; HAN, S. A centroid-based heuristic algorithm for the capacitated vehicle routing problem. *Computing and Informatics*, v. 30, n. 4, p. 721–732, 2012. Citado 3 vezes nas páginas 30, 66 e 69.

SHUKLA, N.; TIWARI, M.; CEGLAREK, D. Genetic-algorithms-based algorithm portfolio for inventory routing problem with stochastic demand. *International Journal of Production Research*, Taylor & Francis, v. 51, n. 1, p. 118–137, 2013. Citado 3 vezes nas páginas 37, 41 e 84.

SOLOMON, M. M. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, Informs, v. 35, n. 2, p. 254–265, 1987. Citado 2 vezes nas páginas 32 e 79.

SOLOMON, M. M.; DESROSIERS, J. Survey paper—time window constrained routing and scheduling problems. *Transportation science*, INFORMS, v. 22, n. 1, p. 1–13, 1988. Citado na página 43.

SOLYALI, O.; CORDEAU, J.-F.; LAPORTE, G. Robust inventory routing under demand uncertainty. *Transportation Science*, INFORMS, v. 46, n. 3, p. 327–340, 2012. Citado 2 vezes nas páginas 37 e 43.

TAN, K. C. et al. Heuristic methods for vehicle routing problem with time windows. *Artificial intelligence in Engineering*, Elsevier, v. 15, n. 3, p. 281–295, 2001. Citado 3 vezes nas páginas 32, 79 e 80.

TOTH, P.; VIGO, D. *The Vehicle Routing Problem*. [S.l.]: Society for Industrial and Applied Mathematics, 2002. (Monographs on Discrete Mathematics and Applications). Citado 2 vezes nas páginas 20 e 32.

TRUDEAU, P.; DROR, M. Stochastic inventory routing: Route design with stockouts and route failures. *Transportation Science*, INFORMS, v. 26, n. 3, p. 171–184, 1992. Citado 2 vezes nas páginas 37 e 38.

WOLSEY, L. A. *Integer programming*. New York, NY, USA: Wiley-Interscience, 1998. Citado 2 vezes nas páginas 26 e 27.

XIAO, N.; RAO, Y. L. Multi-Product Multi-Period Inventory Routing Optimization with Time Window Constrains. *International Journal of Simulation Modelling*, v. 15, n. 2, p. 352–364, jun. 2016. Citado 3 vezes nas páginas 44, 45 e 48.

YU, Y.; CHEN, H.; CHU, F. A new model and hybrid approach for large scale inventory routing problems. *European Journal of Operational Research*, Elsevier, v. 189, n. 3, p. 1022–1040, 2008. Citado 2 vezes nas páginas 22 e 84.

YU, Y.; CHU, F.; CHEN, H. A model and algorithm for large scale stochastic inventory routing problem. In: IEEE. *2006 International Conference on Service Systems and Service Management*. [S.l.], 2006. v. 1, p. 355–360. Citado 3 vezes nas páginas 37, 42 e 84.