ENOCH YANG

ESTUDO DO PROBLEMA DE ROTEIRIZAÇÃO DE VEÍCULOS APLICADO A UMA EMPRESA DO SETOR DE VAREJO

São Paulo
2019

ENOCH YANG

ESTUDO DO PROBLEMA DE ROTEIRIZAÇÃO DE VEÍCULOS APLICADO A UMA EMPRESA DO SETOR DE VAREJO

Trabalho de Formatura apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do diploma de Engenheiro de Produção.

São Paulo
2019

ENOCH YANG

ESTUDO DO PROBLEMA DE ROTEIRIZAÇÃO DE VEÍCULOS APLICADO A UMA EMPRESA DO SETOR DE VAREJO

Trabalho de Formatura apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do diploma de Engenheiro de Produção.

Orientadora:

Prof[a]. Dr[a]. Débora Pretti Ronconi

São Paulo
2019

Catalogação-na-publicação

Yang, Enoch
Estudo do Problema de Roteirização de Veículos Aplicado a uma Empresa do Setor de Varejo / E. Yang -- São Paulo, 2019.
127 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Produção.

1.Pesquisa Operacional 2.Logística 3.Roteirização de veículos 4.Heurística I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Produção II.t.

À minha família, por me ensinar a ser
resiliente e confiar no Senhor

AGRADECIMENTOS

À Deus em primeiro lugar.

Aos meus pais, Yang Junfeng e Zhu Xiaodan, e minha irmã, Eloisa Yang, por todo o amor, incentivo e paciência ao longo dos anos. Sem eles eu não estaria aqui.

À Professora Doutora Débora Pretti Ronconi por me introduzir à Pesquisa Operacional e ao uso de métodos quantitativos em Engenharia, por todo o incentivo e apoio, pela orientação deste presente trabalho e por confiar em mim ao longo da graduação.

Ao Guilherme Mainieri e Tiago Freitas da B2W, pela oportunidade de realizar este trabalho em parceria com a empresa e pelo apoio e colaboração ao longo da elaboração deste.

Aos meus amigos e colegas espalhados pelo mundo, por terem me acompanhado até aqui, partilhando momentos de alegria e de sofrimento.

A todo o corpo docente da Escola Politécnica da Universidade de São Paulo, de outras unidades da USP e da École Polytechnique que contribuíram para a minha formação acadêmica.

"All models are approximations. Assumptions, whether implied or clearly stated, are never exactly true. All models are wrong, but some models are useful. So the question you need to ask is not 'Is the model true?' (it never is) but 'Is the model good enough for this particular application?' "

-George E. P. Box-

RESUMO

Este trabalho tem como objeto de estudo a roteirização para o abastecimento de lojas de uma empresa do setor de varejo, propondo, com a utilização de técnicas de pesquisa operacional, uma roteirização mais eficiente. O problema estudado tem foco nas operações nas regiões Norte e Nordeste do Brasil, regiões em que as lojas são mais esparsas e as distâncias percorridas maiores. Para a otimização do processo de roteirização desenvolveu-se um modelo de programação linear inteira e mista, caracterizando um problema de roteamento de veículos com janelas de tempo, frota heterogênea e períodos de descanso. Por conta do caráter **NP**-difícil do problema, foi necessária a utilização de um método heurístico para buscar uma solução para o problema real. Foi utilizada uma heurística construtiva de inserção, derivada da heurística de Solomon (1987) e obteve-se uma solução de boa qualidade, reduzindo significativamente o custo comparado à solução atual empregada pela empresa. Avalia-se no final do trabalho a robustez do método heurístico obtido a partir de análises de sensibilidade.

Palavras-Chave – Pesquisa Operacional, Logística, Roteirização de veículos, Heurística.

ABSTRACT

This work studies the vehicle routing problem applied to a retail company, and proposing, with the use of Operations Research techniques, a more efficient routing process for the company. The problem studied focuses on the operations in the northern and northeastern regions in Brazil, where the stores distributions are sparse, and the distances travelled are longer. For the optimization of the routing, a mixed integer linear programming model was developed, characterizing a vehicle routing problem with a heterogeneous fleet, time windows and rest times. Because of the **NP**-hard aspect of the problem, the use of heuristic methods was necessary to obtain a solution. We used a constructive insertion heuristic derived from the original Solomon's heuristic (1987), and obtained a good quality solution which significantly reduces the costs for the company. At the end we also evaluate the robustness of the method by submitting it to some sensibility analysis.

Keywords – Operations Research, Logistics, Vehicle routing, Heuristics.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ACS *Adapted Combined Savings*.

AOOS *Adapted Optimistic Opportunity Savings*.

API *Application Programming Interface*.

AROS *Adapted Realistic Opportunity Savings*.

CD Centro de Distribuição.

GS *Grid Search*.

LASA Lojas Americanas S.A..

NP *Non-deterministic Polynomial*.

P *Polynomial*.

PIB Produto Interno Bruto.

PLIM Programação Linear Inteira Mista.

RVRP *Rich Vehicle Routing Problem*.

TP *Transit Point*.

TSP *Traveling Salesman Problem*.

VRP *Vehicle Routing Problem*.

VUC Veículo Urbano de Carga.

SUMÁRIO

INTRODUÇÃO

O presente trabalho busca estudar e melhorar um aspecto da operação logística da B2W Companhia Digital, uma empresa de varejo digital subsidiária da Lojas Americanas S.A. A B2W passa por uma fase de expansão das suas operações de vendas de produtos próprios e de terceiros por meio do chamado *Marketplace*. Nesse contexto, a operação de logística e *fullfilment*[1] se torna uma etapa essencial para a concretização do plano estratégico da empresa.

A operação logística é fundamental para o setor de varejo, em particular para a B2W e a Lojas Americanas S.A., dentro do contexto de sua expansão de lojas e realização do seu plano de criação de uma plataforma multicanal de varejo. A importância das operações logísticas para o grupo é evidenciada pela criação da LET'S, subsidiária dedicada à gestão dos ativos e das operações de logística, no ano de 2018.

Acredita-se que a gestão logística pode agregar valor pela redução de custos de transporte, que são em geral significativos e compõem uma parcela importante dos gastos das empresas de todo o país. Em pesquisa feita pelo ILOS - Instituto de Logística e Supply Chain, o custo da logística no Brasil representa mais de 12% do PIB (ILOS, 2017); e segundo um levantamento da Fundação Dom Cabral, em 2017 o custo logístico das empresas representou em média 12.37% do faturamento bruto destas (FUNDAÇÃO DOM CABRAL, 2017).

Assim, é de interesse da B2W que as diferentes etapas do processo de *fullfilment* sejam realizadas de maneira eficiente e com o menor custo possível. Alguns dos objetivos da LET'S atualmente envolvem a integração e a otimização da malha logística do grupo. Neste sentido, o presente trabalho procura contribuir para este objetivo, com o desenvolvimento de um modelo matemático e de métodos sistemáticos para obter rotas de menor custo, permitindo à empresa atender seus clientes da melhor maneira possível.

Este trabalho aborda especificamente o problema de roteirização de veículos para o abastecimento de lojas da Lojas Americanas S.A. nas regiões Norte e Nordeste do país e está estruturado em 7 capítulos.

O Capítulo 1 expande a descrição da empresa na qual o trabalho será realizado com um breve histórico, sua estratégia, a cadeia de valor do setor e também a relevância da logística

[1] O processo completo desde a busca de informação no ponto de ventda até a entrega do produto ao consumidor

para a empresa.

O Capítulo 2 apresenta o problema a ser resolvido, justificando a sua relevância para a empresa. Expõe em mais detalhes o contexto, a maneira como a empresa ataca o problema atualmente e as suas principais características.

O Capítulo 3 reúne a bibliografia relacionada necessária para a resolução do problema, e em particular sobre o tópico de roteirização de veículos, as diferentes variantes desse problema e os principais métodos que são aplicados para a resolução do problema.

O Capítulo 4 formaliza a metodologia que dirige o estudo e também apresenta os dados necessários para o estudo do problema, junto com a metodologia para a sua obtenção.

O Capítulo 5 desenvolve o modelo matemático do problema, apresentando seus detalhes, implementação e testes computacionais para compreender os seus limites.

O Capítulo 6 descreve a classe de métodos heurísticos, complementares ao modelo matemático, utilizada para resolver o problema. Especifica como os métodos são adaptados para a realidade da empresa e detalhando a performance do método comparado ao método exato.

O Capítulo 7 analisa a solução obtida a partir dos métodos apresentados no trabalho e disserta sobre a sua robustez com relação a mudanças de cenário.

Por fim, o capítulo de conclusão encerra o trabalho. Após a conclusão, o leitor encontrará a lista de referências bibliográficas e os apêndices do trabalho, que incluem um método de geração de dados aleatórios, os códigos desenvolvidos ao longo do trabalho e o roteiro da solução final proposta.

1 DESCRIÇÃO DA EMPRESA

Este capítulo tem como objetivo apresentar a empresa na qual o trabalho será realizado. Primeiramente, apresenta-se o grupo B2W e a sua maior acionista, a Lojas Americanas S.A., junto com a evolução do seu modelo de negócios. Em seguida, apresenta-se a estratégia de expansão do grupo e a iniciativa de logística LET'S, bem como o papel da logística no setor de varejo e mais especificamente dentro do contexto da empresa. Assim, cria-se o contexto adequado para a apresentação do problema no próximo capítulo.

1.1 O grupo B2W Digital

No final de 2006, como resultado da fusão entre as empresas Submarino, Americanas.com e Shoptime, foi criada a B2W Companhia Digital, uma empresa de comércio eletrônico que é hoje líder em comércio eletrônico na América Latina e de relevância em escala mundial.

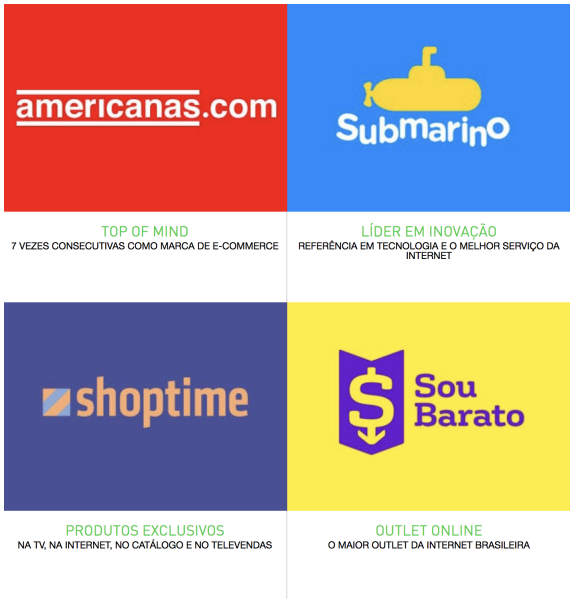
A companhia possui grandes marcas de *e-commerce* como Americanas.com, Submarino, Shoptime e SouBarato, além de presença em outros setores (mas também ligados ao *e-commerce*) como *Marketplaces*, meios de pagamentos, BIT *services* (TI), logística e distribuição, etc. As quatro grande marcas do grupo estão representadas na Figura 1.

A B2W é uma empresa de capital aberto listada na B3, e seu maior acionista é o grupo Lojas Americanas S.A. (LASA), detendo mais de 60% do capital social (posição em 31/08/2018).

1.1.1 Histórico da B2W

- Em 1999, foram criadas a Americanas.com e o Submarino;
- Em 2005, Submarino abre o capital (IPO), adquire o Shoptime e a Ingresso.com;
- Em 2006, cria-se o Submarino Finance, realiza-se uma oferta adicional de ações (follow-on), cria-se o Submarino Viagens. Em Dezembro ocorre a fusão entre Americanas e

Figura 1: Marcas B2W

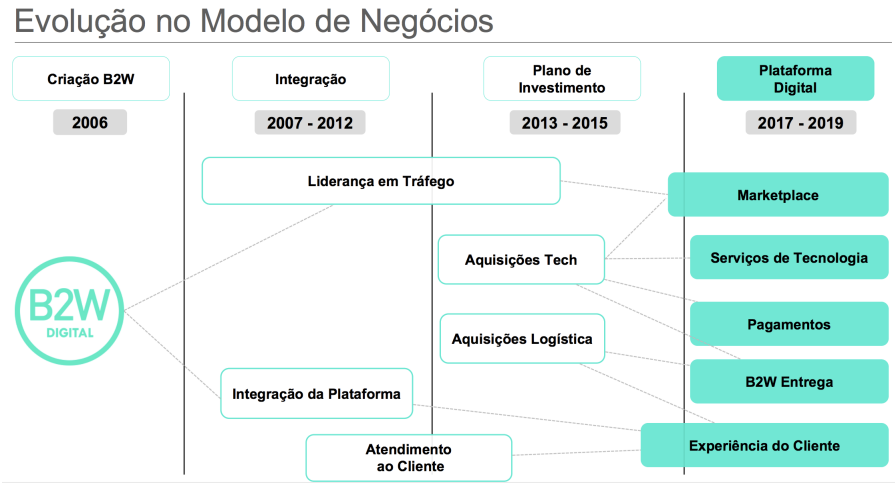


Fonte: Site da empresa (B2W DIGITAL, 2019d)

Submarino, resultando na B2W;

- Em 2007, ocorre a incorporação da B2W com o Shoptime, o grupo adquire o direito de uso da marca Blockbuster na internet e cria-se a B2W Viagens;
- Em 2008, é inaugurado um novo CD da B2W;
- em 2011, ocorre um aumento de capital no valor de 1 bilhão de reais, lançamento do site SouBarato, inauguração do CD da B2W em Recife;
- Em 2012, ocorre a inauguração de 4 novos CDs (RJ, SP, MG e PE);
- Em 2013, mais aquisições (Click Rodo, Uniconsult, Tarkena e Ideais), lançamento do *Marketplace*;
- Em 2014, ocorreu um aumento de capital de 2.38 bilhões de reais, aquisição da Transportadora Direct, inauguração de 4 novos CDs, expansão de 1 CD;
- Em 2015, Lançamento da B2W *Fulfillment*, do cartão SouBarado e do Submarino Prime, aquisição da e-smart, Sieve Group, alienação da B2W viagens e do Ingresso.com;
- Em 2016, ocorre a aquisição da BOO Labs, aumento de capital, lançamento B2W Ads, acordo operacional com a Vialog;
- Em 2017, ocorre um aumento de capital de 1.2 Bilhões, expansão do *Marketplace* e lançamento do Americanas Prime e C2C;

Figura 2: Evolução do plano de negócios da B2W



Fonte: apresentação de resultados B2W 1T19 (B2W DIGITAL, 2019a)

- Em 2018, é realizado o lançamento do AME Digital. As ações BTOW3 são incluídas no Ibovespa.

1.1.2 O estado atual da B2W

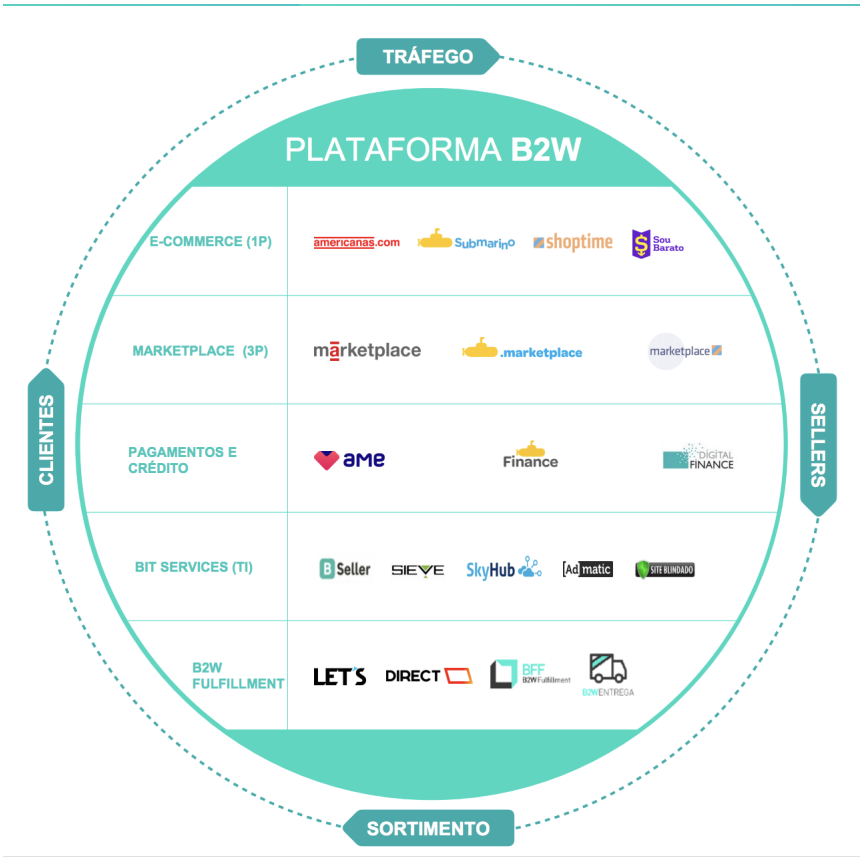
Depois da sua criação em 2006, a B2W passou por uma evolução no seu modelo de negócios que pode ser dividida em três fases: uma primeira fase de integração (de 2007 a 2012); uma segunda fase de investimentos (de 2013 a 2015); e a fase atual, em que ela é uma plataforma digital com várias frentes de atuação integradas entre si. A Figura 2 mostra essa evolução e a Figura 3 mostra a visão da plataforma da B2W, operando o *e-commerce* (venda de produtos próprios) e o *Marketplace* (venda de produtos de terceiros), ambos aliados a serviços diversos tanto para o consumidor quanto para os vendedores na plataforma.

Nos últimos 5 anos (de 2015 a 2019), a B2W vem apresentando crescimento nos seus resultados financeiros e operacionais, resultado de seus investimentos nas suas operações e em diversos projetos de inovação digital.

1.2 Lojas Americanas S.A.

A Lojas Americanas S.A. é uma rede de varejo brasileira, uma das maiores e mais tradicionais do país, com mais de 1320 lojas espalhadas em todo o território. Inaugurada em 1929, a LASA possui um longo histórico de evolução das suas atividades. Nas ultimas décadas, vem apresentando um crescimento contínuo, evidenciado principalmente pelo aumento do número

Figura 3: Plataforma B2W



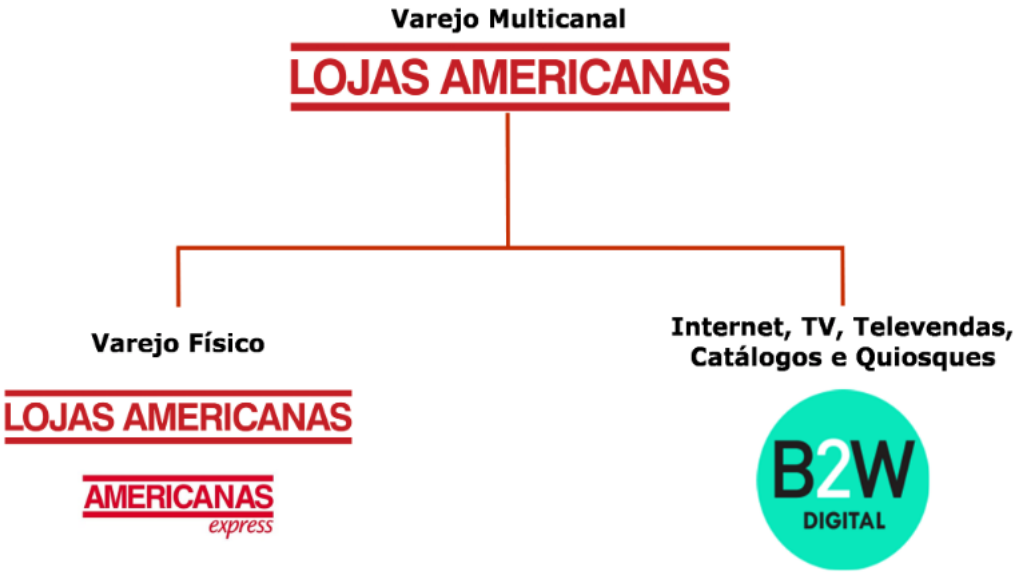
Fonte: apresentação de resultados B2W 1T19 (B2W DIGITAL, 2019a)

de lojas e pela expansão das atividades do grupo B2W, cujo maior acionista é a LASA (como mencionado acima). Pode-se citar por exemplo o plano de expansão "85 Anos em 5 - SOMOS MAIS BRASIL" (LOJAS AMERICANAS S.A., 2019), lançado em 2014, que tinha como objetivo abrir mais de 800 novas lojas e 2 novos Centros de Distribuição entre 2015 e 2019.

Além de suas lojas físicas, a LASA vende produtos no Americanas.com e no *Marketplace*, aumentando a capilaridade de seu alcance através dos diferentes canais. A relação inversa também acontece: a Americanas.com possui também quiosques instalados dentro das Lojas Americanas, oferecendo uma experiência de compra alternativa ao cliente e a comodidade de receber o produto em casa (dentre outros benefícios) e contribuem para a inclusão digital, oferecendo em muitos casos a primeira experiência de compra online do cliente com o auxílio de um associado treinado da loja.

Atualmente a LASA está presente fisicamente em 603 cidades brasileiras; já através do *Marketplace*, atingiu mais de 3500 municípios brasileiros no primeiro trimestre de 2019. O organograma apresentado na Figura 4 ilustra a estrutura da LASA.

Figura 4: Organograma LASA



Fonte: site da LASA (LOJAS AMERICANAS S.A., 2019)

A Lojas Americanas apresenta dois principais formatos de loja: Tradicional e Express. Em seu modelo tradicional, as lojas são amplas (área de venda média de 1000 metros quadrados), com sortimento de até 60 mil itens, separados em mais de 45 departamentos (por exemplo, utilidades domésticas, brinquedos, eletroeletrônicos, higiene pessoal, etc.) e o abastecimento das lojas é diário. No modelo *express*, as lojas se aproximam mais de um formato de loja de con-

veniência, suprindo as necessidades de públicos como os moradores de um determinado bairro. A área de vendas dessas lojas é de 400 metros quadrados em média e cada loja comercializa até 15 mil itens, com logística Just In Time. Na região Sudeste existe também um terceiro tipo de loja, de conveniência, com 100 metros quadrados em média, reposição diária de estoque, sortimento de produtos voltados para conveniência alimentar, comercializando até 3000 itens.

1.3 Estratégia do grupo

Dentro de um ambiente competitivo como o varejo (físico e digital), a LASA e a B2W buscam se destacar com alguns aspectos. Como exemplo, cita-se a formação de uma equipe digital bem sucedida, a criação de uma ampla plataforma de negócios e infraestrutura operacional ao longo dos anos, o que resulta em uma operação multicanal, multi-negócios e multimarcas.

O grupo busca oferecer e garantir a melhor experiência de compra para o cliente, por meio de um "processo simples de compra, uma entrega rápida e eficaz e um melhor serviço de atendimento". A título de exemplo, para determinados produtos e localidades, a infraestrutura logística da B2W permite a realização de entregas no mesmo dia da compra.

O grupo busca crescer com a expansão de lojas físicas (LASA) e da sua presença online (B2W), além de investir em oportunidades ligadas à inovação e serviços. A expansão da plataforma de negócios ocorreu tanto no sentido horizontal quanto no vertical. Por meio de seus 4 sites, a B2W oferece uma grande variedade de produtos em mais de 40 categorias (CD, DVD, livros, games, informática, etc). Por meio de suas subsidiárias, ela oferece também soluções de cartão de crédito, serviços para os vendedores do *Marketplace* (em tecnologia e logística) e operações onlines de grandes marcas por meio do [B]Seller e do B2W *Fulfillment*.

É interessante ressaltar a importância do *Marketplace* dentro do plano estratégico da B2W. A empresa busca se posicionar como uma plataforma digital híbrida que mistura produtos próprios (1P) e de terceiros (3P), além de oferecer serviços relacionados. Essa estratégia vem da visão de que o mercado varejista brasileiro caracteriza-se por lojas com pouco sortimento e pela ausência de grandes *megastores*. Esta deficiência favorece os varejistas online, uma vez que estes não possuem limitação de espaço de prateleira e não necessitam replicar estoques em várias lojas. Assim, o papel do *Marketplace* é de aumentar a variedade da oferta de produtos oferecidos na plataforma, a partir da inclusão de produtos de vendedores independentes de de grandes marcas, permitindo que a B2W ofereça tudo que o cliente procura (*one-stop-shop*) (B2W DIGITAL, 2019c). Essa expansão busca capturar o crescimento do comércio eletrônico no Brasil, que mesmo diante de condições macroeconômicas desafiadoras, apresenta um crescimento im-

Figura 5: Logo da LET'S

Fonte: LET'S

pressionante: em 2018 o faturamento foi de R$ 53,2 bilhões, alta nominal de 12% em relação a 2017, e a projeção para 2019 é de uma expansão de 15%, totalizando R$ 61,2 bilhões em vendas, também segundo dados da eBit-Nielsen (EBIT-NIELSEN, 2019). Este crescimento é impulsionado pela expansão da base de usuários de internet e pelo crescimento do número de consumidores online.

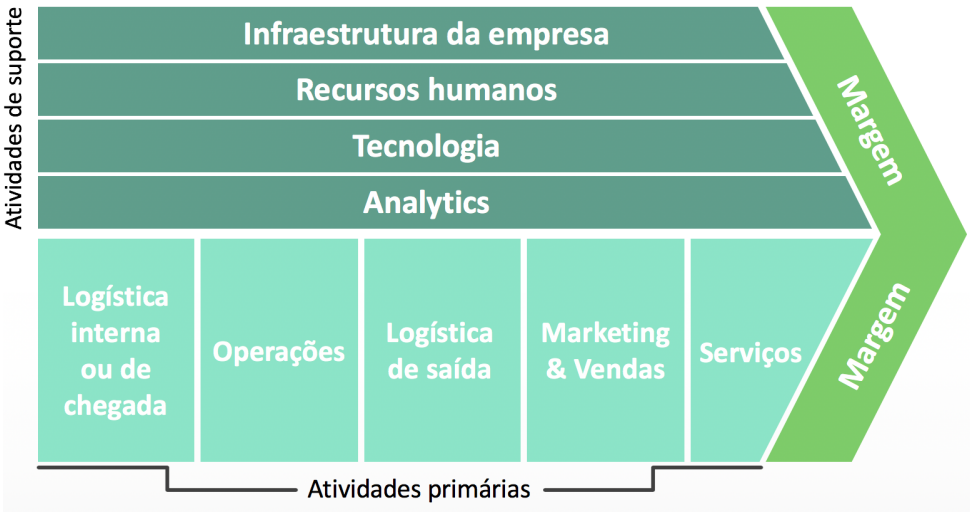
1.4 LET'S - Logística e Distribuição

A integração das atividades da LASA e da B2W não se limita ao elo final, que é a venda ao consumidor. Os dois grupos possuem uma sinergia muito grande no que se trata dos seus ativos logísticos. Diante dessa oportunidade, foi criada em 2018 a *LET'S - Logística e Distribuição*, uma empresa controlada pelos dois grupos, que atua como uma plataforma de gestão compartilhada dos ativos de logística e distribuição da LASA e da B2W. Seu logo é apresentado na Figura 5. A LET'S possui uma estrutura separada dos dois grupos, possuindo até um CEO próprio (VALOR ECONÔMICO, 2019).

Os ativos de logística dos dois grupos cobrem uma grande área do Brasil, por meio de uma malha logística de centros de distribuição, *hubs* (*transit points*), transportadoras e lojas físicas (da LASA), o que torna a LET'S um *player* bastante relevante na logística do Brasil. A criação da LET'S busca aumentar a eficiência logística das duas companhias e também permite aos investidores do mercado avaliar melhor o valor desses ativos.

A atuação da LET'S vai, então, além das empresas do grupo B2W, realizando projetos para a LASA e também projetos de integração entre os dois grupos. A integração da distribuição da LASA com a B2W ainda não acontece de maneira plena, mas está sendo estudada e desenvolvida. Em alguns casos a distribuição da B2W e da LASA é completamente segregada: a título de exemplo, existem Centros de Operações (CO's) da LASA e da B2W na mesma localidade geográfica, mas as operações são completamente independentes, o que sinaliza um espaço para

Figura 6: Cadeia de valor



Fonte: elaborado pelo autor

melhoria, mas também é um grande desafio.

1.5 Cadeia de Valor no varejo

Segundo (PARENTE, 2000) e (RODRIGUEZ *et al.*, 2015), o varejo envolve produtos ou serviços ao consumidor final e é o último estágio de agregação de valor em um canal de distribuição, atuando como um importante elo entre a indústria e o consumidor final. A B2W é uma empresa de varejo digital (*e-Commerce*) e a sua cadeia de valor (na descrição clássica de Porter) inclui os seguintes elos (Figura 6):

- Logística de entrada e logística interna
- Operações
- Logística externa
- *Marketing* e vendas
- Serviços

Além desses elos, a B2W também desenvolve paralelamente outras linhas de negócio, que possuem sinergias com a sua linha de negócios principal. As iniciativas de *Marketplace* permitem à B2W uma diversificação maior dos produtos oferecidos, e os vendedores podem usufruir de uma parte da infraestrutura que a B2W oferece (incluindo suporte técnico e logístico). A

B2W também possui empresas ligadas a pagamentos, serviços digitais e logística, numa ótica de criar uma plataforma unificada e com cobertura completa da cadeia de valor. A diversificação vertical da B2W é uma de suas principais vantagens, essencial para competir em um cenário mundial bastante competitivo com concorrentes americanos e chineses.

1.6 O papel da distribuição

Dentro da ótica da cadeia de valor, a logística tem como objetivo disponibilizar o produto certo no momento e local corretos. Assim, cabe à logística auxiliar o varejo na busca de uma gestão adequada e constante processo de revisão de nível de serviço ao cliente, com o mínimo custo possível (RODRIGUEZ *et al.*, 2015). Segundo (BALLOU, 2003), o valor da logística é manifestado em termos de tempo e localização, já que um produto ou serviço só possui valor se estiverem na posse de um cliente quando e onde ele pretende consumi-lo. Ainda segundo (BALLOU, 2003), alguns grandes desafios na logística são:

- **Custos significativos**: os gastos com logística são significativos e compõem uma parcela importante dos gastos das empresas. Em pesquisa feita pelo ILOS - Instituto de Logística e Supply Chain, o custo da logística no Brasil representa mais de 12% do PIB (ILOS, 2017). Assim, a distribuição pode agregar valor pela minimização desses custos.
- **Expectativas do serviço logístico**: com a Internet, procedimentos operacionais *just-in-time* e reposição de estoques contínua, espera-se o que o processamento de pedidos seja cada vez mais ágil, que a entrega seja imediata e que um produto tenha alta disponibilidade.
- **Complexidade das linhas de suprimento e distribuição**: com uma economia mundial e integrada, existe uma tendencia em que as cadeias de suprimento são estendidas, tornando-se mais complexas e mais custosas.

Sendo um setor estratégico e muito importante para as empresas do setor, a B2W possui uma conjunto de iniciativas que apontam nessa direção - a começar pela criação da LET'S, que reforça a necessidade e o desejo de aprimorar o serviço logístico do grupo. Outro grupo de iniciativas, sob o conceito de "Tudo. A toda hora. Em qualquer lugar.", denominadas O2O - *Online to Offline*, opera com maior grau de integração com as lojas, permitindo ao cliente facilidades como a entrega das compras online nas lojas da LASA (*Click and Collect*) ou a compra online de produtos da loja física da LASA mais próxima. A B2W também oferece o B2W Entrega, operado pela LET'S, um serviço que combina competências em tecnologia, logística

e distribuição que serve o *Marketplace*, reduzindo os prazos de entrega para vendedores terceiros. Recentemente, esta empresa passou a oferecer também o programa O2O para vendedores do *Marketplace*.

A LET'S busca otimizar as operações das companhias por meio de um modelo flexível de *Fulfillment*. Ela é responsável também pela redução dos prazos de entrega da B2W. Hoje a B2W oferece variadas modalidades de entrega, sendo 50% das compras realizadas e enviadas pela LET'S entregues em até 2 dias. De acordo com (B2W DIGITAL, 2019b), a B2W oferece hoje as seguintes modalidades de entrega:

- ***Click & Collect Now***: produto separado e disponível para retirada na loja em 1 hora.
- ***Click & Collect***: entregas na loja em até 48 horas (principais capitais)
- ***2 Hours***: entregas em até 2 horas (capitais SP e RJ)
- ***Same Day***: entregas em até 8 horas (capitais SP e RJ)
- ***Next Day***: entregas em até 24 horas (capitais SP e RJ)
- ***2 Days***: entregas em até 48 horas (capitais SP, RJ, MG, PR, SC, RS e PE)
- ***Standard***: entregas em até 7 dias (Brasil)

Outros projetos que merecem destaque são

- Lançamento de uma plataforma de *crowdshipping*[1] que pode acelerar o processo de entrega para clientes a partir das lojas físicas das Lojas Americanas ou das lojas físicas de vendedores do *Marketplace*.
- Testes de entregas com *Drones*[2]. A B2W é a primeira de varejo brasileiro a realizar voos experimentais para entregas com *Drones*. A expectativa é de realizar entregas de produtos em rotas dos CD's para lojas físicas da Lojas Americanas.

Alguns dos desafios enfrentados pela LET'S atualmente envolvem a integração e a otimização da malha logística do grupo. Os objetivos podem ser separados em projetos de integração e também em projetos de otimização das estruturas já existentes. No Capítulo 2 apresentaremos o problema tratado nesse trabalho e como ele se insere nesse contexto.

[1] entregas por entregadores independentes em modais como motos e bicicletas
[2] veículo aéreo não tripulado e controlado remotamente

2 DESCRIÇÃO DO PROBLEMA

Este capítulo tem como objetivo apresentar o problema a ser resolvido neste trabalho. Após a descrição da empresa e do papel da logística dentro do modelo de negócios da mesma realizado no Capítulo 1 , iniciamos este capítulo descrevendo o contexto atual da empresa, de suas operações de distribuição e o seu interesse na otimização dessas operações. Em seguida, descrevemos o problema que será objeto de estudo nesse trabalho, especificando o objetivo, os requerimentos e as restrições operacionais encontradas, de forma a definir de maneira clara o escopo do problema.

2.1 Evidências da necessidade e do interesse de otimizar a distribuição

Como evidenciado no final do capítulo anterior, a distribuição tem um papel fundamental na cadeia de valor do varejo. Como não há uma transformação fundamental na natureza do produto (contrário a uma indústria em um setor mais tradicional, por exemplo), uma das vantagens competitivas de uma empresa de varejo está na rapidez e eficiência da entrega do produto ao cliente. Além disso, a distribuição representa uma parcela considerável dos custos das empresas do setor. Segundo um levantamento da Fundação Dom Cabral, em 2017 o custo logístico das empresas no estudo (uma amostra de empresas de diversos setores) representou em média 12.37% do faturamento bruto destas (FUNDAÇÃO DOM CABRAL, 2017). Aplicando a mesma proporção, estima-se que a LASA teve um gasto de aproximadamente R$ 1 bilhão no primeiro semestre de 2019; e, considerando o faturamento consolidado (LASA + B2W) de R$ 9.5 bilhões no mesmo período, percebe-se que a otimização da operação logística do grupo pode gerar economias significantes.

2.2 Roteirização do abastecimento de lojas LASA

Atualmente, o planejamento e a roteirização das lojas da LASA são realizados pela própria equipe de operações, baseados na experiência da equipe, sem a utilização de métodos objetivos para a tomada de decisão. Assim, a LET'S entende que existe uma oportunidade de melhor explorar o processo de criação de rotas, buscando otimizar a roteirização das entregas de maneira objetiva e sistemática.

No caso da LASA, o processo de abastecimento das lojas se dá da seguinte maneira:

1. A logística de entrada é realizada pelos fornecedores da LASA. Cada fornecedor (desde eletrônicos a produtos alimentícios) entrega os pedidos nos CD's regionais da LASA.

2. A partir dos CD's, as mercadorias podem ser agregadas em um *Transit Point* (TP) (uma espécie de *hub*). Esta etapa pode ser dispensada a depender da região: uma grande parte das entregas é feita diretamente entre CD's e lojas.

3. A partir dos TP's, as mercadorias são entregues às lojas.

Uma particularidade das lojas da LASA em todo o país é que os pedidos são determinados pela matriz (no Rio de Janeiro), então de modo geral a "produção" é empurrada para as diferentes lojas do território - os gerentes das lojas não escolhem o que terão em estoque, com poucas exceções para lojas maiores em regiões mais densamente populadas.

A LASA possui lojas espalhadas por todo o país, mas a distribuição geográfica das lojas não é uniforme. Na região Sudeste (particularmente, São Paulo e Rio de Janeiro) existe um grande número de lojas próximas aos grandes centros urbanos. Não é o caso em geral nas outras regiões do país, especialmente nas regiões Norte e Nordeste: nestas, o número de lojas é bastante reduzido - consequentemente, a densidade de lojas também, dada a grande extensão em área dessas regiões.

As lojas no Norte e Nordeste são servidas por um único CD, por meio de entregas diretas do mesmo às lojas. A dificuldade encontrada nessas regiões, dada a baixa densidade de lojas, consiste no fato de as mesmas serem esparsas e as distâncias a serem percorridas serem longas; na maior parte dos casos, isto implica em trajetos de duração de vários dias úteis para cada veículo. Isso cria um desafio adicional para a roteirização por conta do horizonte prolongado de planejamento: a frota de veículos não retorna ao CD no final do dia, como seria o caso em regiões onde os clientes são mais próximos. Ainda, há que se levar em conta restrições legislativas com relação à duração da jornada de trabalho dos motoristas, especialmente a Lei

conhecida vulgarmente como “Lei dos caminhoneiros” (BRASIL, 2015). Essas dificuldades podem também ser encontradas em outras regiões do país com distribuições geográficas de lojas semelhantes, como a região do estado de Minas Gerais (MG).

2.3 O problema

O problema a ser resolvido consiste na tarefa de roteirização de veículos para o abastecimento das lojas da LASA nas regiões Norte e Nordeste (tarefa que possivelmente pode se estender para o estado de MG). Este problema se insere no escopo de atuação da LET’S, com o objetivo de melhorar a utilização de recursos logísticos das empresas do grupo (LASA e B2W).

As lojas e o CD nas regiões Norte e Nordeste podem ser visualizados no mapa da Figura 7. Apresentamos também o mapa para o estado de Minas Gerais, na Figura 8. Os ícones em vermelho são os Centros de Distribuição e os pontos restantes são as lojas.

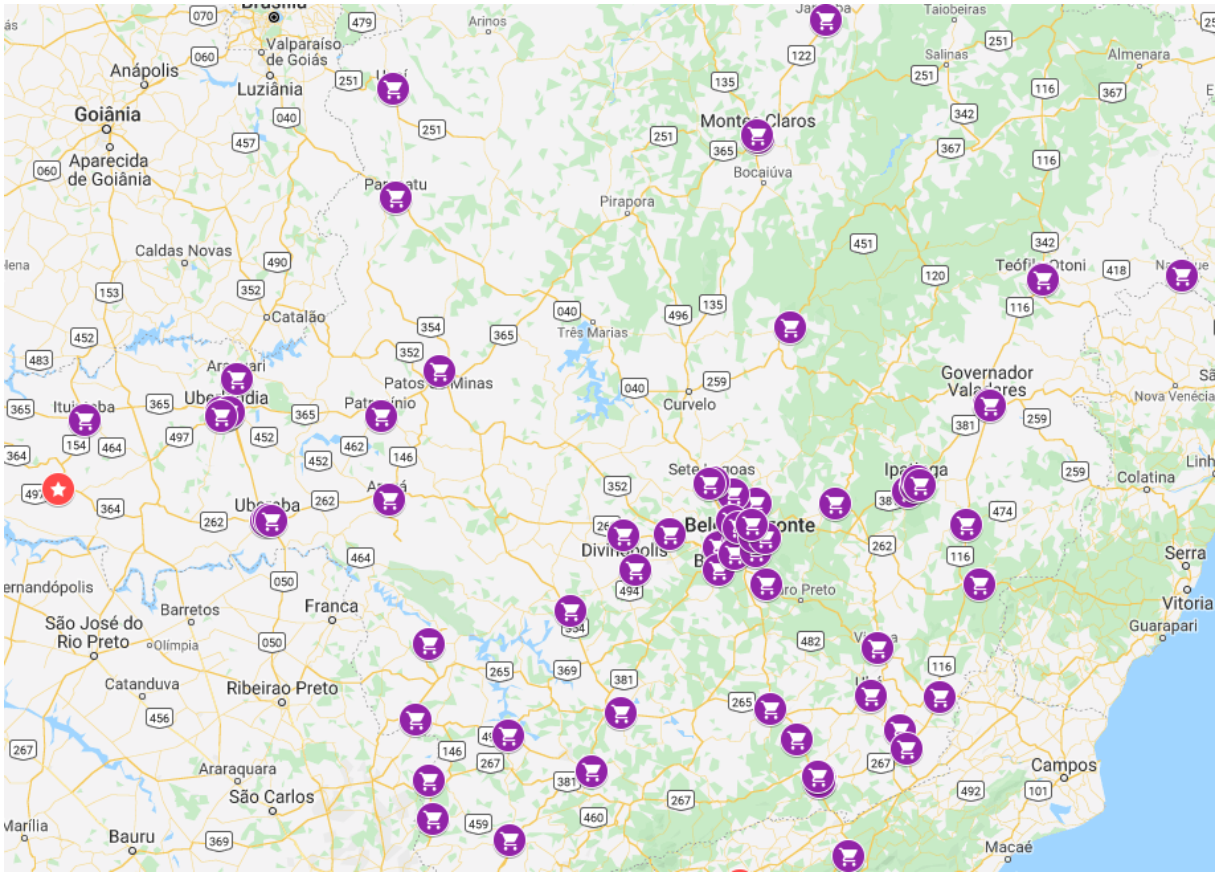
Figura 7: Lojas e CD, regiões Norte e Nordeste

Fonte: Elaborado pelo autor com auxílio do serviço *Google Maps*

As regiões Norte e Nordeste concentram ao todo 28% das lojas da LASA, totalizando 376 lojas (281 no Nordeste e 95 no Norte), atrás apenas da região Sudeste que concentra mais de 50% das lojas. Estes números podem ser constatados na Figura 9. O CD que serve ambas regiões está localizado em Recife/PE, apresentado na Figura 10.

Como mencionado no capítulo anterior, além das Lojas Americanas “tradicionais”, a LASA

Figura 8: Lojas e CD, estado de Minas Gerais

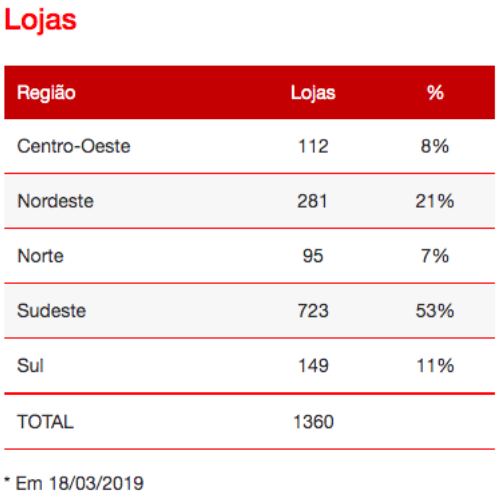


Fonte: Elaborado pelo autor com auxílio do serviço *Google Maps*

possui outros dois tipos de lojas: Americanas *Express* e Local Americanas. Além dessa diferenciação, cada tipo de loja tem dimensões e contextos diferentes (por exemplo, podem ser lojas de rua ou lojas de *Shopping Center*'s). Essas especificidades geram restrições espaço-temporais para cada loja que precisam ser respeitadas. As lojas possuem tamanhos e configurações diferentes, o que limita o tipo de veículo que pode atendê-las. Ainda, há que se levar em consideração o cronograma de cada loja: estas possuem janelas de entregas variáveis, definidas em função do dia da semana em que ocorre a entrega (apenas uma janela por dia). Por exemplo, lojas localizadas em *Shopping Center*'s possuem janelas de entrega bastante restritas, normalmente no período da manhã. Finalmente, cada loja possui uma frequência diferente de abastecimento (de 1 a 6 dias por semana) de acordo com suas capacidades de estoque e a demanda a ser atendida; esta frequência também é definida pela matriz da LASA no Rio de Janeiro. As janelas e as frequências são previamente fixadas em um dado horizonte de planejamento.

As entregas são realizadas por transporte rodoviário. O grupo possui uma frota de veículos própria, operada pela transportadora Direct; em caso de necessidade, o excedente é terceirizado, buscando-se transportadores disponíveis. Existem três tipos de veículo: *truck*'s, carretas

Figura 9: Distribuição de lojas LASA por região



Lojas

Região	Lojas	%
Centro-Oeste	112	8%
Nordeste	281	21%
Norte	95	7%
Sudeste	723	53%
Sul	149	11%
TOTAL	1360	

* Em 18/03/2019

Fonte: Site da empresa (LOJAS AMERICANAS S.A., 2019)

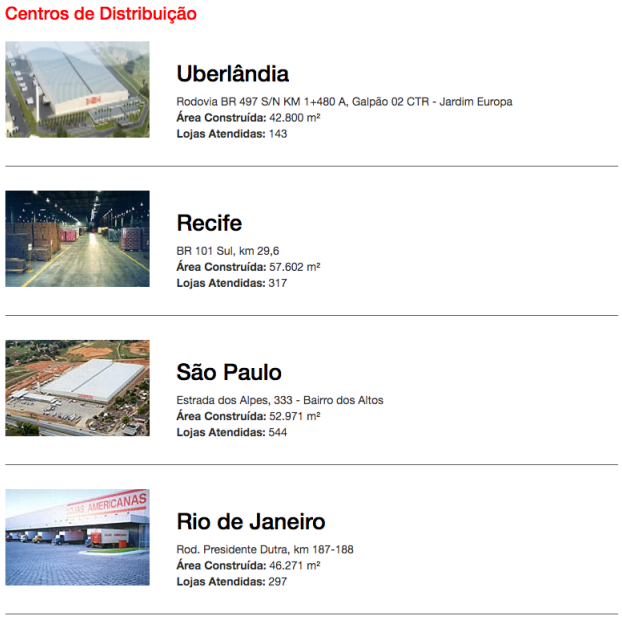
e caminhões 3/4's (também conhecidos como Veículo Urbano de Carga (VUC)), com diferentes capacidades, velocidades e custos de operação (custo fixo e frete) para cada tipo.

Outras restrições operacionais são:

- Cada veículo está sujeito a uma limitação no valor total das cargas (limitando possíveis perdas), na sua capacidade volumétrica e também no peso total do carregamento (por conta de regulamentações rodoviárias). As entregas da LASA tendem a saturar as restrições de volume ou peso.
- A duração máxima da jornada de trabalho do motorista deve ser respeitada. Devido às grandes distâncias entre lojas e entre as lojas e o CD, o tempo de percurso de uma entrega pode ultrapassar o limite de horas de trabalho consecutivas de um dado motorista (8 horas), o que implica a necessidade de mais um dia de rota.
- Além da restrição da jornada máxima de trabalho dos motoristas, há também uma restrição de tempo mínimo de descanso intrajornada. São necessárias 16 horas de descanso a cada 1 dia (24 horas).
- Os tempos de carregamento e descarregamento são significativos, então precisam ser levados em conta na roteirização.

Diante do cenário descrito acima, a LET'S tem então como objetivo buscar um roteiro que minimize os custos de transporte (que variam em função de custo fixo por dia e por veículo e frete por quilômetro rodado) e que atenda a demanda e os prazos de abastecimento das lojas,

Figura 10: Localização dos CD's LASA



Fonte: Site da empresa (LOJAS AMERICANAS S.A., 2019)

respeitando as demais restrições supramencionadas (capacidade de peso e volumétrica, valor máximo, jornada de trabalho, tipo de veículo aceito e janelas de tempo por loja).

2.3.1 Exemplo de uma rota viável

Para ilustrar o problema descrito, especialmente o aspecto das janelas de tempo e de tempo de descanso, apresentamos o seguinte exemplo simplificado. As localidades das lojas, os respectivos prazos, janelas de entrega e restrições de tipo de veículo estão descritas na Tabela 1. Para os dados fornecidos, uma roteirização viável/factível para um veículo de tipo "Truck" e

Tabela 1: Dados para o exemplo de problema

Localidade	Prazo	Janela	Tipos de veículo
Loja A	D2	Manhã (09:00-12:00)	Truck, Carreta
Loja B	D3	Tarde (12:00-17:00)	Truck
Loja C	D1	Manhã	Truck
Loja D	D2	Manhã	Truck, Carreta
Loja E	D3	Tarde	Truck, Carreta
Loja F	D2	Tarde	Truck
Loja H	D3	Manhã	Truck, Carreta
Loja I	D3	Tarde	Truck

Fonte: B2W

um motorista é data na Tabela 2. Nesse exemplo supomos que as demandas são atendidas e as capacidades volumétricas e de valor são respeitadas. Cada linha da Tabela 2 representa uma

Tabela 2: Roteiro possível para o exemplo com veículo Truck

Dia	Origem	Destino	Início Origem	Fim Origem	Início Destino	Fim Destino
D1	CD	Loja A	9	10	10,5	11
D1	Loja A	Loja B	10,5	11	13	14
D1	Loja B	Loja C	13	14	16,5	16,5
D2	Loja C	Loja D	9	10	11	12
D2	Loja D	Loja E	11	12	14,5	15,5
D2	Loja E	Loja F	14,5	15,5	16	17
D3	Loja F	Loja H	9	9	10	11
D3	Loja H	Loja I	10	11	14	15
D3	Loja I	CD	14	15	16	16

Fonte: B2W

viagem a ser realizada. A primeira linha refere-se a uma viagem planejada para o dia D1, partindo do CD para a Loja A, com instante de início do carregamento na origem à 09:00 e fim às 10:00 (1h de carregamento), tempo de percurso de 00:30 e finalmente um descarregamento de meia hora.

É importante notar que as jornadas de trabalho de 8h são respeitadas em todos os dias de rota e também o intervalo de descanso de 16h entre os dias (D1-D2 ou D2-D3). O abastecimento da loja C neste caso ilustra uma viagem que ocorre em um dia e o descarregamento no dia seguinte, visto que o tempo de descarregamento violaria a restrição da jornada de trabalho do motorista no dia D1.

O problema descrito acima pertence a uma classe de problemas conhecida como Problemas de Roteamentos de Veículos. No Capítulo 3 apresentaremos uma revisão da bibliografia acerca desse problema.

3 REVISÃO BIBLIOGRÁFICA

Este capítulo será consagrado à revisão bibliográfica acerca dos temas pertinentes ao problema a ser resolvido, mais especificamente da literatura em torno do problema de roteirização de veículos e suas extensões, como introduzido no Capítulo 2. O objetivo do capítulo é reunir o fundamento teórico necessário para desenvolver a solução para o problema.

Inicia-se o capítulo apresentando o problema de roteamento de veículos, a sua formulação matemática e a sua extensão com janelas de tempo. Em seguida classifica-se o problema a ser resolvido segundo uma taxonomia, sistematizando assim as principais características do problema relevantes para a sua resolução. Por fim, o capítulo termina com uma apresentação dos métodos de resolução existentes para esta classe de problemas, passando antes por uma breve discussão sobre complexidade computacional.

3.1 Problemas de Roteamento de veículos

O problema estudado nesse trabalho pertence a uma classe de problemas conhecida na literatura como Problemas de Roteamentos de Veículos - *Vehicle Routing Problem* (VRP). Na sua forma mais clássica, o VRP é um problema de otimização inteira e combinatória, em que temos um ou mais depósitos e um conjunto de clientes formando uma malha, um conjunto de veículos, um produto e a sua demanda pelos respectivos clientes. O objetivo do problema é encontrar um conjunto de rotas de menor distância que visite todos clientes, satisfazendo as suas demandas (e eventuais restrições operacionais). Este é um problema central para a gestão de logística e precisa ser resolvido rotineiramente por transportadoras (LAPORTE, 2009).

O VRP surgiu como uma generalização do famoso Problema do Caixeiro Viajante - *Traveling Salesman Problem* (TSP), onde um vendedor busca encontrar a menor rota para percorrer um conjunto de cidades (visitando uma única vez cada uma delas) e por fim retornando à cidade de origem. Uma das primeiras descrições do VRP é dada em (DANTZIG; RAMSER, 1959), sob o nome de *Truck Dispatching Problem*, diferente do nome pelo qual o problema é conhecido hoje.

Desde então, várias extensões do problema foram propostas, incluindo novos elementos na função objetivo (minimização de custos, da distância percorrida, do número totalde veículos, do tempo total de atendimento, etc.) ou novas restrições e aspectos operacionais ao problema. Alguns exemplos clássicos de extensões do problema de roteamento de veículos (VRP) são:

- VRP com múltiplos depósitos, onde mais de um depósito está disponível, então veículos podem partir de qualquer um deles para atender os clientes.
- VRP periódico, onde o horizonte de planejamento é de vários dias (ao invés de um dia). Aqui o número de entregas para cada cliente é um dado do problema; portanto, além das rotas, precisa-se decidir em quais dias dentro do horizonte cada cliente deve ser visitado. Por exemplo, em um horizonte de 6 dias e uma frequência de 3 entregas pode-se escolher os dias 1, 3, e 5 ou 2, 4, e 6 para as entregas (ou qualquer outra combinação de dias).
- VRP com entregas fracionadas (DROR; LAPORTE; TRUDEAU, 1994), onde a demanda de um cliente pode ser satisfeita por mais de um veículo. Isso ocorre na situação em que um primeiro veículo pode entregar uma fração da demanda total do cliente e depois outro veículo entrega o restante, o que permite rotas mais flexíveis ou uma taxa de utilização melhor para os veículos.
- VRP estocásticos: são uma classe ampla de VRP's em que uma ou várias partes do problema são de natureza aleatória (por exemplo, clientes aleatórios, demandas aleatórias, tempos aleatórios) (LAPORTE; LOUVEAUX, 1998).
- VRP com janelas de tempo, aonde os clientes possuem janelas de tempo em que podem receber entregas, de forma que os veículos precisam chegar no cliente em um horário dentro dessas janelas.
- VRP com coletas e entregas, em que os veículos devem, além de entregar, coletar produtos nos clientes.

3.1.1 Formulação do VRP clássico

A formulação matemática do problema de roteamento de veículos clássico é essencial para a busca de uma solução ótima e também para compreender as formulações dos problemas estendidos, então apresentamos a seguir uma possível formulação de Programação Linear Inteira Mista (PLIM) segundo a descrição em (ARENALES *et al.*, 2015).

Uma malha logística, com um CD (ou depósito), clientes e rotas que os interligam, pode ser representada por um grafo $G = (N, E)$, em que $N = C \cup \{0, n+1\}$, onde $C = \{1, \ldots, n\}$

é o conjunto de nós que representa os clientes e o restante são os nós de origem e destino final (ambos representando o CD). O conjunto $E = \{(i, j) : i, j \in N, i \neq j, i \neq n+1, j \neq 0\}$ corresponde aos arcos entre os nós. Note que nenhum arco termina no nó 0 e nenhum arco começa no nó $n+1$. Essa separação do depósito em dois nós, um como origem e outro como destino final, é conveniente, como se verá adiante. Todas as rotas devem começar em 0 e terminar em $n+1$. Um custo c_{ij} e tempo de trajeto t_{ij} são associados a cada arco $(i, j) \in E$. Aqui o tempo de viagem t_{ij} inclui o tempo de serviço do cliente i. Cada cliente i tem uma demanda d_i. Um conjunto K de veículos idênticos está disponível e situado no depósito, e cada veículo $k \in K$ tem capacidade Q.

O objetivo é encontrar um conjunto de rotas que atenda a todos os clientes, satisfazendo suas demandas e que minimize o custo total de viagens, sujeito às restrições:

- Cada rota inicia e termina no depósito.
- Cada cliente pertence somente a uma rota.
- A demanda total de uma rota não pode exceder a capacidade Q do veículo.
- O tempo de viagem de uma rota não pode exceder o limite D.

Definimos as variáveis binárias

$$x_{ijk} = \begin{cases} 1, & \text{se o veículo } k \text{ percorre o arco } (i, j),\ \forall k \in K,\ \forall (i, j) \in E \\ 0, & \text{caso contrário} \end{cases}$$

e temos então a formulação:

$$\min_{x} \quad \sum_{k \in K} \sum_{(i,j) in E} c_{ij} x_{ijk} \tag{3.1a}$$

$$\text{sujeito a} \quad \sum_{k \in K} \sum_{j \in N} x_{ijk} = 1, \qquad \forall i \in C \tag{3.1b}$$

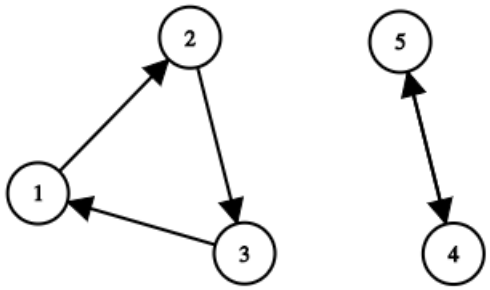
$$\sum_{i \in C} d_i \sum_{j \in N} x_{ijk} \leq Q, \qquad \forall k \in K \tag{3.1c}$$

$$\sum_{i \in N} \sum_{j \in N} t_{ij} x_{ijk} \leq D, \qquad \forall k \in K \tag{3.1d}$$

$$\sum_{j \in N} x_{0jk} = 1, \qquad \forall k \in K \tag{3.1e}$$

$$\sum_{i \in N} x_{ihk} = \sum_{j \in N} x_{hjk}, \qquad \forall h \in C, \forall k \in K \tag{3.1f}$$

Figura 11: Exemplo de sub-rotas



Fonte: elaborado pelo autor

$$\sum_{i \in N} x_{i,n+1,k} = 1, \qquad \forall k \in K \qquad (3.1g)$$

$$\sum_{i \in S} \sum_{j \in S} x_{ijk} \leq |S| - 1,\ S \subset C,\ 2 \leq |S| \leq \lfloor \frac{n}{2} \rfloor, \qquad \forall k \in K \qquad (3.1h)$$

$$\mathbf{x} \in B^{|K||E|} \qquad (3.1i)$$

A equação (3.1a) é a função objetivo, representando o custo total das rotas. A restrição (3.1b) impõe que cada cliente só é visitado uma única vez. As restrições (3.1c) e (3.1d) restringem a capacidade de cada veículo e o tempo total de cada rota (veículo). As restrições (3.1e), (3.1f) e (3.1g) são comumente chamadas de restrições de fluxo: os veículos devem sair do depósito (0) e terminar no depósito $(n+1)$; e se um veículo chega em um cliente h, deve sair desse mesmo cliente h. A restrição (3.1i) impõe que as variáveis x sejam binárias (0 ou 1).

Finalmente a restrição (3.1h), menos intuitiva em primeira vista, tem suas origens na formulação e resolução do Problema do Caixeiro Viajante em (DANTZIG; FULKERSON; JOHNSON, 1954). Conhecida como restrição de eliminação de sub-rotas, essa restrição impede a formação de circuitos fechados e isolados nos roteiros. Um exemplo de solução do TSP contendo sub-rotas é mostrado na Figura 11, com 5 nós. Aqui a restrição funciona da seguinte maneira: impondo a cada veículo que o número de arcos entre os clientes de qualquer subconjunto $S \in C$ tem que ser menor ou igual ao número de clientes desse conjunto menos 1 (ou seja, $|S| - 1$), geometricamente temos que o número de arestas (rotas) não é suficiente para completar o polígono que seria formado pelos vértices (clientes), efetivamente impedindo a formação de circuitos isolados. A verificação dessa restrição para todos os subconjuntos $S \in C$ tem natureza combinatória, o que implica uma complexidade computacional muito elevada. O número de subconjuntos que podem ser construídos a partir de um conjunto de n elementos é 2^n, um calculo que torna-se rapidamente proibitivo quando o número n de clientes é grande.

3.1.2 Roteamento de veículos com janelas de tempo

Na extensão com janelas de tempo do VRP, para cada cliente i o início do serviço deve estar dentro da janela de tempo $[a_i, b_i], i \in C$. Os veículos deixam o depósito no instante 0 e devem retornar durante o intervalo $[a_{n+1}, b_{n+1}]$. Aqui consideramos que um veículo pode chegar em um nó antes da sua janela de tempo e esperar sem custo. Essa formulação também é retirada de (ARENALES *et al.*, 2015).

Para modelar a característica de tempo do problema, consideramos a variável adicional:

$$s_{ij} = \text{instante em que o veículo } k \text{ começa a servir o cliente } i,\ \forall k \in K,\ \forall i \in C$$

Consideramos também o seguinte conjunto de restrições adicionais: se o veículo k deixa o nó i e viaja para o nó j, então não pode chegar em j antes de $s_{ik} + t_{ij}$:

$$x_{ijk}(s_{ij} + t_{ij} - s_{jk}) \leq 0,\ \forall (i,j) \in E,\ \forall k \in K. \tag{3.2}$$

Note que essa restrição de tempo não é linear, pois temos o produto entre variáveis de decisão, mas ela pode ser linearizada considerando um grupo de constantes $M_{ij} = \max\{b_i + t_{ij} - a_j, 0\}$:

$$s_{ik} + t_{ij} \leq s_{jk} + (1 - x_{ijk})M_{ij},\ \forall (i,j) \in E,\ \forall k \in K \tag{3.3}$$

Finalmente, todas as janelas de tempo devem ser respeitadas:

$$a_i \leq s_{ik} \leq b_i,\ \forall i \in N - \{0\},\ \forall k \in K \tag{3.4}$$

Adicionando as restrições (3.3) e (3.4) ao VRP, temos a formulação do problema de roteamento de veículos com janelas de tempo. É interessante notar que com a adição da restrição (3.3) ao problema, as restrições de eliminação de sub-rotas (3.1h) não são mais necessárias, segundo (ARENALES *et al.*, 2015).

3.2 Taxonomia de classificação de problemas de roteamento de veículos

Como mencionado acima, existem várias extensões possíveis ao problema de roteamento de veículos, e muitas delas foram objeto de estudo e pesquisa nos últimos anos. Os autores em (EKSIOGLU; VURAL; REISMAN, 2009) e (BRAEKERS; RAMAEKERS; NIEUWENHUYSE, 2016), após uma revisão da literatura sobre o roteamento de veículos, criam uma taxonomia bastante detalhada de variantes do problema de roteamento de veículos. No entanto,

essa lista se torna rapidamente extensa ao tentar incorporar todos os possíveis detalhes e variantes dos problemas existentes na literatura. Em (LAHYANI; KHEMAKHEM; SEMET, 2015), os autores criam uma taxonomia um pouco mais abrangente e geral de uma classe de problemas denominada *Rich Vehicle Routing Problem* (RVRP), ou problemas de roteamento de veículos "ricos", no sentido de que esses problemas possuem uma variedade maior de restrições, motivadas por problemas de roteamento "da vida real" segundo os autores. A taxonomia é apresentada abaixo:

1. Características de cenário
   - (a) Dados de entrada
     - i. Estáticos
     - ii. Dinâmicos
     - iii. Determinísticos
     - iv. Estocásticos
   - (b) Componentes de decisão
     - i. Roteamento
     - ii. Inventário e roteamento
     - iii. Localização e roteamento
     - iv. Roteamento e *scheduling* (programação) de motoristas
     - v. Planejamento da produção e distribuição
   - (c) Número de depósitos
     - i. Único
     - ii. Múltiplos
   - (d) Tipo de operação
     - i. Coleta ou entrega
     - ii. Coleta e entrega
     - iii. *Backhaul* (retornos)
     - iv. *Dial-a-ride* (restrições de serviço)
   - (e) Fracionamento de cargas
     - i. Permitido
     - ii. Não permitido
   - (f) Período de planejamento
     - i. único
     - ii. múltiplo
   - (g) Uso de veículos
     - i. Viagem única
     - ii. Múltiplas viagens
2. Características físicas
   - (a) Veículos
     - i. Tipo
       - A. Homogêneo
       - B. Heterogêneo
     - ii. Número
       - A. Fixo
       - B. Ilimitado
     - iii. Estrutura
       - A. Compartimentalizada
       - B. Única
     - iv. Restrições de capacidade
     - v. Política de carregamento
       - A. Ordem cronológica
       - B. Sem política
     - vi. Regulações dos motoristas
   - (b) Restrições de tempo
     - i. Restrições no cliente

ii. Restrições na rota
iii. Restrições no depósito
iv. Tempo de serviço
v. Tempo de espera

(c) Estrutura de janelas de tempo

i. Janela única
ii. Múltiplas janelas

(d) Restrições de incompatibilidade

(e) Restrições específicas

(f) Função objetivo

i. Objetivo único
ii. Múltiplos objetivos

O problema a ser resolvido nesse trabalho pode ser classificado segundo essa taxonomia, e a classificação é apresentada na Tabela 3.

Tabela 3: Classificação do problema segundo a taxonomia de RVRP's

Características do cenário	
Dados de entrada	Determinísticos
Componentes de decisão	Roteamento
Número de depósitos	Depósito único
Tipo de operação	Entrega apenas
Restrições de repartição da carga	Fracionamento não permitido
Período de planejamento	Período único (estático)
Uso múltiplo de veículos	Viagem única
Características físicas do problema	
Tipo de frota	Heterogênea
Tamanho da frota	Ilimitada
Restrições de capacidade	Massa, volume e valor
Política de carregamento	Sem política
Regulações sobre os motoristas	Tempo máximo de trabalho por dia e tempo mínimo de descanso por dia
Restrições de tempo	No cliente, no depósito, tempo de serviço
Janelas de tempo	Janela de tempo única
Restrições de incompatibilidade	Compatibilidades entre veículos e clientes
Função objetivo	Objetivo único

Fonte: elaborado pelo autor

3.3 Métodos de resolução

A resolução dos problemas de roteirização de veículos pode ser exata ou heurística. Nos métodos exatos, a solução encontrada é garantidamente a solução ótima do problema. No entanto, o seguinte problema aparece: o problema de roteirização de veículos pertence à classe

de complexidade **NP**-*hard* (**NP**-difícil em português) (LENSTRA; KAN, 1981), brevemente descrita abaixo.

3.3.1 Uma breve discussão sobre complexidade

A descrição e discussão completa do conceito da classe de complexidade NP foge do escopo desse trabalho, pois é um domínio de pesquisa bastante vasto em complexidade computacional. O que segue é inspirado na descrição dada por (PAPADIMITRIOU; STEIGLITZ, 1982).

A complexidade de tempo de um algoritmo é o tempo/número de operações necessário para executar o algoritmo em função do tamanho dos dados de entrada. Também fala-se de complexidade de espaço, que é o tamanho de memória necessário para executar o algoritmo, também em função do tamanho dos dados de entrada. Normalmente essas complexidades são expressas com a notação O-grande (ou notação Landau), que representa o comportamento assintótico de uma função quando o seu argumento tende a infinito ou a algum valor específico. Por exemplo, se o número de operações de um certo algoritmo para uma entrada de tamanho n é de $9n^2 + 2n$, então sua complexidade computacional é quadrática: $O(n^2)$ pois o termo em n^2 é dominante quando n tende a infinito. Normalmente um algoritmo é considerado eficiente se ele tem complexidade polinomial, ou seja se o seu tempo de execução é majorado por $O(n^k)$, k constante.

A classe de complexidade **P** (*Polynomial*) inclui todos os problemas de decisão (um problema cuja resposta é sim ou não) que podem ser resolvidos em tempo polinomial (existe um algoritmo, cuja complexidade é polinomial, que resolve o problema). Já a classe **NP** (*non-deterministic polynomial time*) inclui os problemas de decisão em que, dada uma solução cuja resposta é "sim", esta pode ser verificada em tempo polinomial. Claramente temos que $\mathbf{P} \subseteq \mathbf{NP}$, no entanto não se sabe ainda se $\mathbf{P} = \mathbf{NP}$. Dentro da classe **NP**, existe um subconjunto de problemas chamados de **NP**-completos, problemas de decisão que são os problemas "mais difíceis" em **NP**, e todos os problemas em **NP** podem ser transformados ou reduzidos em tempo polinomial a algum problema em **NP**-completo. Duas características interessantes dos problemas **NP**-completo são:

1. Atualmente não se conhece nenhum algoritmo polinomial que possa resolver um problema **NP**-completo.

2. Se existir um algoritmo polinomial que resolve um único problema **NP**-completo, então ele resolve todos os problemas **NP**-completos.

E finalmente os problemas **NP**-difícil são aqueles que são pelo menos "tão difíceis quanto"

os problemas em **NP**-completo (existe uma redução polinomial de um problema em **NP**-completo para um problema em **NP**-difícil) e não são necessariamente problemas de decisão.

De maneira geral, um problema de otimização não é um problema de decisão, pois a resposta não se resume a um sim ou um não. No entanto, pode-se reformular um problema de otimização para que ele tenha uma versão que seja um problema de decisão. Por exemplo, se temos o seguinte problema de genérico de otimização:

"Dada uma instância do problema, com os conjuntos de dados S (para verificar se uma solução é factível) e Q (para calcular o custo $c(f)$ de uma solução f), encontre a solução ótima."

A sua versão de decisão pode ser descrita por

"Dada uma instância do problema, com os dados S e Q e um inteiro L, existe uma solução factível f tal que o seu custo $c(f)$ seja menor ou igual a L?"

Note a diferença na natureza da resposta que cada um dos dois problemas exige. O primeiro exige uma solução factível ótima, enquanto o segundo deve ser respondido com um sim ou não. Deste modo, na literatura é comum classificar os problemas de otimização cuja versão de decisão é **NP**-completa, dentro da classe **NP**-difícil (PAPADIMITRIOU; STEIGLITZ, 1982).

Para o presente trabalho a consequência prática do fato de um problema ser **NP**-difícil (que é o caso do VRP) é que não se conhece um algoritmo de complexidade polinomial para resolver o problema, e qualquer algoritmo que resolva corretamente o problema requer, no pior caso, um tempo exponencial, tornando inviável a sua resolução exata quando o tamanho das instâncias (número de clientes) aumenta.

3.3.2 Métodos de resolução de VRP's

Em (TOTH; VIGO, 2014) os autores fazem uma revisão dos principais métodos publicados para a resolução do VRP e suas variantes.

Os métodos exatos de resolução são aqueles utilizados para a resolução de problemas de programação inteira e mista, com destaque aos métodos do tipo *branch and bound* ou *branch and cut*. Na sua forma mais genérica, o algoritmo de *branch and bound* é um algoritmo de busca em problemas discretos e de otimização combinatória (como no caso do VRP). O algoritmo inicia enumerando o espaço de soluções, transformando-o numa estrutura de árvore, e realiza a busca seguindo os ramos da arvore. Antes de se aprofundar em um ramo, este é comparado

com estimativas de limitantes superiores e inferiores da solução ótima, e caso o ramo não possa produzir soluções melhores, ele é descartado, limitando um pouco o espaço de busca.

Problemas de otimização combinatória sofrem do problema do crescimento exponencial do espaço de busca: se todas as n variáveis forem binárias, o espaço de busca teria tamanho 2^n. A estimativa de melhores e mais eficientes limitantes superiores e inferiores pode então melhorar bastante a performance do método, mas ainda assim suas limitações persistem.

Métodos heurísticos são métodos que podem fornecer soluções razoáveis ao problema a ser resolvido, baseando a busca de soluções em hipóteses simplificadoras com o objetivo de mitigar a dificuldade computacional do problema e garantir uma complexidade de tempo razoável (polinomial), mas sacrificando a qualidade da solução encontrada para tal: não existe a garantia de que uma solução encontrada pela heurística é a solução ótima. É dizer: esta pode ser um ótimo local, mas não se sabe se é um ótimo global. Listamos a seguir algumas classes de heurísticas propostas para variantes do VRP:

- Heurísticas construtivas: buscam criar soluções factíveis do zero, podendo fornecer uma solução inicial para outros métodos heurísticos. Podemos citar por exemplo a heurística de *savings* de Clark & Wright, a heurística de inserção de Solomon, algoritmos de varredura e algoritmos de pétalas.
- Heurísticas de melhorias: dada uma solução (factível ou infactível), buscar melhorá-la a partir de movimentos intra-rotas ou entre-rotas.
- Algoritmos em duas fases (*Cluster & Route*): nesse tipo de heurística, criam-se grupos de clientes e em seguida resolvemos um problema de roteamento separado para cada grupo de clientes.
- Meta-heurísticas
  - Busca local: métodos que exploram o espaço de soluções buscando uma solução "vizinha" à solução atual que melhore a função objetivo. Exemplos incluem: *Simulated Annealing*, busca tabu, *Deterministic Annealing*, *Iterated Local Search*, *Variable Neighborhood Search*
  - Algoritmos populacionais: métodos que a partir de uma população inicial de soluções tentam combiná-las para gerar soluções melhores. Exemplos incluem: Colônia de formigas, algoritmos genéticos
- Métodos híbridos: nessa classe de métodos, uma combinação dos métodos citados acima

é utilizada (combinações entre heurísticas, combinações entre heurísticas e métodos exatos)

4 METODOLOGIAS

Após a realização da revisão bibliográfica no Capítulo 3, que dá ao presente trabalho fundamento teórico, este capítulo é dedicado à descrição da metodologia a ser aplicada no estudo. O capítulo inicia apresentando uma metodologia de resolução de problemas ligados à pesquisa operacional. Em seguida apresenta-se uma descrição dos dados necessários para o estudo e a metodologia para o levantamento destes.

4.1 Metodologia de resolução do problema

Segundo (WINSTON; GOLDBERG, 2004), ao utilizar pesquisa operacional para resolver um problema de uma organização, deve-se seguir o seguinte processo em sete passos:

1. **Formular o problema**: o pesquisador define o problema da organização, o que inclui especificar os objetivos da organização e identificar as partes da organização que precisam ser estudadas antes de resolver o problema.

2. **Observar o sistema**: em seguida, o pesquisador deve coletar dados para estimar os parâmetros que afetam o problema da organização. Essas estimativas serão utilizadas nas etapas seguintes para desenvolver e avaliar o modelo matemático do problema.

3. **Formular o modelo matemático**: nesse passo, o pesquisador desenvolve o modelo matemático que representa o problema, utilizando técnicas matemáticas e de pesquisa operacional para modelagem de sistemas.

4. **Verificar e utilizar o modelo para predições**: o pesquisador agora busca determinar se o modelo desenvolvido no passo anterior é uma representação precisa da realidade utilizando métodos de validação do modelo. Mesmo se o modelo for válido para o cenário atual, é necessário aplicá-lo com cautela, já que novas restrições podem surgir conforme o ambiente muda.

5. **Selecionar uma alternativa viável**: dado o modelo e as soluções que o modelo propõe, o pesquisador deve escolher aquela que melhor se enquadra nos objetivos da organização.

6. **Apresentar os resultados**: nessa etapa o pesquisador deve apresentar o modelo e a recomendação da etapa 5 para o grupo tomador de decisão. Em algumas situações, várias alternativas podem ser apresentadas, de forma que cabe à organização escolher aquela que melhor responde às suas necessidades. Após apresentar os resultados do estudo, é possível que a organização não aceite a recomendação, resultado de uma definição incorreta do problema ou da falta de envolvimento do tomador de decisão desde o começo do projeto. Neste caso deve-se retornar às etapas 1, 2 ou 3.

7. **Implementar e avaliar as recomendações**: se a organização aceitar o estudo, então o pesquisador deve ajudar na implementação das recomendações. O sistema deve ser constantemente monitorado (e atualizado dinamicamente conforme o ambiente mudar) para garantir que as recomendações permitem à organização atingir os seus objetivos.

4.2 Metodologia de coleta e descrição dos dados

Os dados para o problema foram fornecidos pela LET'S e seguem um formato pré-determinado pela empresa. Por serem dados estratégicos ligados à operação da empresa, uma camada extra de confidencialidade é adicionada: os dados são sanitizados, ou seja, não é possível identificar as localidades reais das lojas e das rotas realizadas.

A princípio, assumimos que as tabelas contêm todos os dados necessários para o problema, evitando uma fase de inferência sobre os dados. Eles são apresentados em quatro tabelas:

- Localidades,
- Capacidades,
- Cargas,
- Malha.

A seguir, descrevemos cada uma delas e detalhes da metodologia para obtê-las.

A tabela denominada "Localidades" contém a lista de lojas (e o CD), as respectivas janelas de tempo e o tempo de serviço (tempo de carga e tempo de descarga). O formato está apresentado na Tabela 4 (com números fictícios). Existe uma diferenciação dos tempos de carga e

descarga, mas como no problema consideramos somente carga no CD e descarga nas lojas, apenas uma das colunas será utilizada dependendo da localidade. As janelas de tempo são dadas em horas e os tempos de serviço são dados em minutos. As janelas de tempo são fornecidas pelas próprias lojas, e nos dados já serão apresentadas somadas ao dia da entrega. Por exemplo, para uma entrega no segundo dia e uma janela de tempo entre 8 e 11 horas no período da manhã, a janela de tempo representada nos dados será de 32 até 35 horas, pois $8+24\times(2-1)=32$ e $11+24\times(2-1)=35$. Os tempos de carga e descarga são estimados a partir de dados históricos da empresa.

Tabela 4: Formato da planilha de Localidades, números simulados

Tipo	Localidade	Início	Fim	Tempo Carga	Tempo Descarga
BASE	CD	0	96	78	0
DESTINO	LOJA 1	30	34	0	168
DESTINO	LOJA 2	36	42	0	115

Fonte: B2W

A tabela denominada "Capacidades" contém a lista de tipos de veículos, as respectivas capacidades em volume, custos fixos, frete, valor máximo transportado e tamanho da frota por veículo. O formato está apresentado na Tabela 5 (com números fictícios). Decidiu-se trabalhar apenas com capacidades volumétricas (em m^3), facilmente adaptável aos modelos; o custo fixo é dado em Reais; e o frete em R$/km. Os tipos de veículos disponíveis, as suas respectivas capacidades e os valores da tabela de frete foram obtidos pela transportadora Direct, subsidiária da B2W. O tamanho da frota é estimado de acordo com o número de caminhões e motoristas que a empresa está habituada a contratar.

Tabela 5: Formato da planilha de Capacidades, números simulados

Veículos	Capacidades	Custo Fixo	Max Qt Veículos	Frete Km	Máximo Ticket
Veículo 1	28	90	12	1.15	109,000.00
Veículo 2	42	126	14	1.65	109,000.00

Fonte: B2W

Em seguida, temos a tabela denominada "Cargas", que contém a lista de entregas a serem feitas, com a loja, o dia da entrega, o volume a ser entregue e o valor. O formato está apresentado na Tabela 6 (com números fictícios). Ao invés de fornecer uma relação de produtos, suas características (massa, volume e valor) e a demanda por cada produto em cada loja, decidiu-se fornecer o volume e valor agregados de cada entrega. Além disso, decidiu-se omitir o peso de cada entrega. Caso uma loja tenha mais de uma entrega (em dias diferentes) no horizonte de

tempo do planejamento, ela aparecerá duas vezes nessa tabela. Como descrito no Capítulo 2, os pedidos de cada loja (as entregas) são determinadas diretamente de maneira centralizada pela matriz no Rio de Janeiro da LASA, ao invés de serem gerados por cada loja.

Tabela 6: Formato da planilha de Cargas, números simulados

Loja	Dia	Volume	Valor
Loja 1	1	25.81	17,237.79
Loja 2	2	10.26	6,174.25

Fonte: B2W

Finalmente, temos a tabela denominada "Malha", que contém a lista de arcos da malha, contendo a origem, o destino, a distância (em km), o tempo (em horas) e o tipo de veículo associado. O formato está apresentado na Tabela 7 (com números fictícios). A informação do veículo associado ao arco informa a compatibilidade entre lojas e veículos, ou seja, se um tipo de veículo não é aceito pela loja, não existirão arcos chegando ou partindo dessa loja com o tipo de veículo não aceito. As distâncias foram estimadas pela LET'S, os tempos são calculados assumindo uma velocidade média de 60 km/h para os veículos, considerando que os trajetos terão um *mix* de trechos em rodovias e trechos urbanos.

Tabela 7: Formato da planilha de Malha, números simulados

Origem	Destino	Veículo	Distância	Tempo
CD	Loja 1	Veículo 1	100	1.25
Loja 1	Loja 2	Veículo 1	80	1

Fonte: B2W

Para uma noção das dimensões reais do problema, apresentamos algumas características dos dados reais na Tabela 8.

Tabela 8: Dimensões do problema real

Característica	Dimensão
Lojas	107
Tipos de veículos	3
Tamanho total da frota	43
Horizonte de planejamento	12 dias
Distância máxima de um arco	2.317 km
Distância média de um arco	689 km
Janela de tempo média	6.6 horas

Fonte: B2W

5 MODELAGEM DO PROBLEMA

Este capítulo é dedicado ao detalhamento do processo de modelagem do problema, desenvolvido com base no referencial teórico do Capítulo 3 e na metodologia descrita no Capítulo 4.

Inicialmente, uma descrição objetiva do problema e o vocabulário necessário são introduzidos; em seguida, o modelo é apresentado sob a forma de um problema de programação linear inteira mista, junto com a explicação mais detalhada das restrições associadas. Em seguida, apresentamos o modelo aplicado a uma instância do problema de tamanho reduzido, mostrando o comportamento do modelo.

Após a apresentação do exemplo, passamos pela descrição de alguns detalhes sobre a implementação do modelo exato utilizando o *software* IBM ILOG CPLEX; por fim, apresentamos uma série de testes computacionais para compreensão dos limites da resolução exata do problema, o que acaba apontando para a necessidade de buscar soluções heurísticas, tema do próximo capítulo.

5.1 Considerações sobre o problema

A partir de um único CD serão atendidas C lojas por horizonte de planejamento. Todas as lojas possuem frequência unitária, ou seja, são visitadas apenas uma vez. Para isso, lojas que possuem mais de uma entrega planejada são consideradas lojas diferentes (entregas diferentes). A natureza dos dados permite que essa segregação seja feita, pois as quantidades a serem entregues a cada vez e o número de entregas são pré-determinados; por exemplo, uma loja que possui 3 entregas na semana (segunda, quarta e sexta) é dividida em três lojas, e cada uma terá a janela de tempo $[a_i, b_i]$ correspondente ao dia da semana e ao cliente.

Todos os roteiros partem do CD e cada roteiro serve uma sequência de lojas, sem repetições. Cada loja i possui uma demanda d_{ip} para cada produto p dentre uma gama de $|P|$ produtos no total e essa demanda é inteiramente servida por um único veículo (não há repartição da demanda entre veículos). O tempo de serviço (descarregamento) é u_i. A frota de veículos é

heterogênea: dois tipos de veículos próprios e um terceiro tipo terceirizado. O conjunto K de veículos contém toda a frota, ordenada segundo o tipo de veículo: $K = K^1 \cup K^2 \cup K^3 = \{k_1^1, k_2^1, \ldots, k_1^2, k_2^2, \ldots, k_1^3, k_2^3, \ldots\}$. Os veículos próprios são representados por K^1 e K^2 e a frota terceirizada é representada por K^3. A frota total em teoria é ilimitada, pois o que excede a frota própria pode ser terceirizado, mas consideramos um limitante superior de $|C|$ veículos para a frota terceirizada, pois na situação limite, cada cliente é servido por no máximo um veículo, logo $|K^3| = |C|$. Cada tipo de veículo possui uma capacidade de massa, de volume e de valor que pode carregar.

Para cada tipo de veículo existe um conjunto de arcos E^k que definem uma malha de distâncias (d_{ij}^k) e tempos (t_{ij}^k) específica. A malha de distâncias reflete as restrições dos diferentes veículos referentes aos trechos/clientes que eles podem acessar (se um arco não existe em E^k, significa que o veículo não pode acessar esse trecho). Os tempos refletem as velocidades dos veículos nos arcos (d_{ij}^k) e também não serão computados caso o arco não exista.

O veículo deve chegar no cliente dentro da janela de tempo $[a_i, b_i]$ especificada e essa janela não inclui o tempo de serviço. Cada veículo possui um limite no número de horas consecutivas de trabalho por dia e um período de descanso mínimo obrigatório após a jornada de trabalho.

Buscamos uma solução (um conjunto de rotas viáveis) que atenda às demandas e restrições dos clientes e veículos, e que minimize a soma dos custos fixos (por veículo utilizado) e custos variáveis (frete, proporcional à distância percorrida). Temos então um problema de programação linear inteira mista PLIM de roteamento de veículos com janelas de tempo, pausas obrigatórias e frota heterogênea e ilimitada.

5.2 Modelagem do problema a ser resolvido

Considere então os seguintes elementos:

Conjuntos:

- C é o conjunto de lojas a serem atendidas
- K é o conjunto de veículos
- P é o conjunto de produtos

Dados

Associados ao produto p

- m_p massa do produto
- v_p volume do produto
- r_p valor do produto

Ligados a um cliente i

- d_{ip} demanda de produto p
- $[a_i, b_i]$ janela de tempo associada
- u_i é o tempo de descarregamento

Associados a um veículo k

- CM_k restrição de peso
- CV_k restrição de volume
- CR_k restrição de valor
- cf_k custo fixo de utilização do veículo
- cv_k custo variável (frete) do veículo
- e_k hora de início de trabalho

Associados ao arco (i, j)

- d_{ij}^k distância do trajeto para o veículo k
- t_{ij}^k tempo de trajeto para o veículo k

Variáveis:

$s_{ik} =$ instante do início do atendimento do cliente i pelo veículo $k, \forall k \in K, \forall i \in C \cup \{0\}$

$$x_{ijk} = \begin{cases} 1, & \text{se o veículo } k \text{ percorre o arco } (i,j), \forall k \in K, \forall (i,j) \in E^k \\ 0, & \text{caso contrário} \end{cases}$$

$$y_k = \begin{cases} 1, & \text{se o veículo } k \text{ é utilizado}, \forall k \in K \\ 0, & \text{caso contrário} \end{cases}$$

$$z_{ik} = \begin{cases} 1, & \text{se o veículo } k \text{ realiza o descanso após visitar i}, \forall k \in K \\ 0, & \text{caso contrário} \end{cases}$$

Apresenta-se então o modelo de PLIM. No que segue, ε é um número real suficientemente pequeno, e M', M'' são números inteiros suficientemente grandes.

$$\min_{x} \quad \sum_{k\in K} \sum_{(i,j)\in E^k} cv_k d_{ij} x_{ijk} + \sum_{k\in K} cf_k y_k \tag{5.1a}$$

subject to

$$\sum_{k\in K} \sum_{j\in N-\{0,i\}} x_{ijk} = 1, \qquad \forall i \in C \tag{5.1b}$$

$$\sum_{i\in C} (\sum_{p\in P} d_{ip} m_p) \sum_{j\in N-\{0,i\}} x_{ijk} \leq CM_k y_k, \qquad \forall k \in K \tag{5.1c}$$

$$\sum_{i\in C} (\sum_{p\in P} d_{ip} v_p) \sum_{j\in N-\{0,i\}} x_{ijk} \leq CV_k y_k, \qquad \forall k \in K \tag{5.1d}$$

$$\sum_{i\in C} (\sum_{p\in P} d_{ip} r_p) \sum_{j\in N-\{0,i\}} x_{ijk} \leq CR_k y_k, \qquad \forall k \in K \tag{5.1e}$$

$$\sum_{j\in N-\{0,i\}} x_{0jk} = 1, \qquad \forall k \in K \tag{5.1f}$$

$$\sum_{i\in N-\{h\}} x_{ihk} = \sum_{j\in N-\{h\}} x_{hjk}, \qquad \forall h \in C, \forall k \in K \tag{5.1g}$$

$$\sum_{i\in N-\{n+1\}} x_{i,n+1,k} = 1, \qquad \forall k \in K \tag{5.1h}$$

$$a_i \leq s_{ik} \leq b_i, \qquad \forall i \in C, \forall k \in K \tag{5.1i}$$

$$s_{ik} + u_i \leq b_i, \qquad \forall i \in C, \forall k \in K \tag{5.1j}$$

$$s_{0k} = e_k, \qquad \forall k \in K \tag{5.1k}$$

$$s_{ik} + t_{ij} + u_i \leq s_{jk} + (1 - x_{ijk}) M_{ij}, \qquad \forall (i,j) \in E, \forall k \in K \tag{5.1l}$$

$$s_{ik} + u_i - e_k = n^s_{ik} D + w^s_{ik}, \qquad \forall (i,j) \in E, \forall k \in K \tag{5.1m}$$

$$0 \leq w^s_{ik} \leq D - \varepsilon, \qquad \forall i \in C, \forall k \in K \tag{5.1n}$$

$$w^s_{ik} + t_{ij} - a_d + \varepsilon \leq M'_{ik}(1 - z_{ik}) + M''(1 - x_{ijk}), \qquad \forall (i,j) \in E, \forall k \in K \tag{5.1o}$$

$$w^s_{ik} + t_{ij} = n^t_{ijk} a_d + w^t_{ijk}, \qquad \forall (i,j) \in E, \forall k \in K \tag{5.1p}$$

$$0 \leq w^t_{ijk} \leq a_d - \varepsilon, \qquad \forall (i,j) \in E \tag{5.1q}$$

$$(s_{ik} + u_i + t_{ij} + n^t_{ijk} t_d - s_{jk})$$
$$\leq M'_{ik} z_{ik} + M''(1 - x_{ijk}), \forall (i,j) \in E, \forall k \in K \quad (5.1r)$$
$$y_k = \sum_{j \in C} x_{0jk}, \quad \forall k \in K \quad (5.1s)$$
$$\mathbf{x} \in B^{|K||E|} \quad (5.1t)$$
$$\mathbf{y} \in B^{|K|} \quad (5.1u)$$
$$\mathbf{s} \in \mathbb{R}_+^{|C||K|} \quad (5.1v)$$
$$n^s_{ik} \in \mathbb{N}^{|C||K|} \quad (5.1w)$$
$$n^t_{ijk} \in \mathbb{N}^{|C|^2|K|} \quad (5.1x)$$

5.2.1 Descrição do modelo

A equação (5.1a) é a função objetivo, que é a soma dos custos variáveis e fixos da rota. Os custos variáveis são dados pelo frete multiplicado pela distância percorrida nos arcos escolhidos e o custo fixo é a soma dos custos fixos dos veículos utilizados.

As restrições (5.1b) garantem que cada cliente seja visitado exatamente uma única vez.

As restrições (5.1f), (5.1g) e (5.1h) são restrições ligadas ao fluxo de veículos e garantem respectivamente que as rotas saiam do depósito onde os veículos são carregados, que elas sejam contínuas e que retornem ao depósito no final da rota. A noção de continuidade representada pela restrição (5.1g) funciona da seguinte maneira: se um veículo k entra no nó h vindo de algum nó i (logo, $x_{ihk} = 1$ para algum i), ele necessariamente precisa sair do nó h indo para algum nó j ($x_{hjk} = 1$ para algum j).

As restrições (5.1c), (5.1d) e (5.1e) representam as restrições de capacidade (massa, volume e valor) do veículo k. A soma da demanda de cada produto para todos os clientes servidos pelo veículo, multiplicadas pela quantidade correspondente (massa, volume, valor) não pode ultrapassar as respectivas capacidades do veículo.

As restrições (5.1i), (5.1j), (5.1k) e (5.1l) pertencem à formulação do problema de roteamento de veículos com janelas de tempo. O início do serviço de cada cliente deve acontecer dentro da respectiva janela de tempo segundo a restrição (5.1i) e o serviço deve terminar antes do fim da janela de tempo em (5.1j). A restrição (5.1k) fixa o início dos veículos no CD no tempo 0. A restrição (5.1l) estabelece a continuidade do serviço de uma rota: se o cliente j é servido imediatamente após o cliente i pelo mesmo veículo k, o horário de início do atendimento do cliente j deve necessariamente ser maior ou igual ao horário de início do atendimento em i, acrescido do tempo de serviço em i e o tempo de trajeto entre i e j.

Para lidar com o problema do tempo de descanso, foram criadas as restrições (5.1m), (5.1n), (5.1o), (5.1p), (5.1q) e (5.1r), que serão detalhadas com maior enfase abaixo.

A restrição (5.1s) define a relação entre x e y: um veículo é utilizado ($y_k = 1$) se ele sai do depósito para algum cliente j. Por fim as restrições restantes (5.1t), (5.1u), (5.1v), (5.1w) e (5.1x) definem os domínios das variáveis.

5.2.2 Construção da restrição de descanso

A condição de tempo máximo de trabalho e tempo mínimo de descanso por dia de cada veículo é uma característica que diferencia o problema em questão de um VRP com janelas de tempo mais clássico. No que segue, detalhamos o processo de construção das restrições ligadas ao descanso dos veículos. Para isso, utilizaremos da seguinte estrutura de restrições "se - então", retirada de (WINSTON; GOLDBERG, 2004). Suponha $f(x_1, x_2, \ldots, x_n)$ e $g(x_1, x_2, \ldots, x_n)$ funções reais, e que queremos garantir que $f > 0 \implies g \geq 0$, então pode-se utilizar a seguinte formulação:

$$-g(x_1, x_2, \ldots, x_n) \leq My \tag{5.2}$$

$$f(x_1, x_2, \ldots, x_n) \leq M(1 - y) \tag{5.3}$$

onde M é um inteiro positivo tal que $f \leq M$ e $-g \leq M, \forall x_1, x_2, \ldots, x_n$, e y é uma variável de decisão binária. Nessa construção, se $f > 0$, então necessariamente $y = 0$ pela segunda equação, e então na primeira equação $g \geq 0$. A contrapositiva ($g < 0 \implies f \leq 0$) pode ser verificada da mesma maneira. Temos também que se $f \leq 0$, a variável y pode tomar tanto o valor 0 quanto 1 e então as restrições não são ativadas; logo, essa estrutura de restrições cumpre exatamente a função desejada, além de manter uma estrutura linear.

Utilizaremos também um caso especial dessa estrutura de restrições, numa estrutura que chamaremos aqui de restrições indicatriz[1], onde queremos utilizar uma variável binária para ativar uma restrição. Suponha $b \in \{0, 1\}$ uma variável binária e uma função $f(x_1, x_2, \ldots, x_n)$, e a condição a ser expressada é: se $b = 0 \implies f \leq 0$. Isso pode ser interpretado da seguinte maneira: a restrição $f(x_1, x_2, \ldots, x_n) \leq 0$ só deve ser ativada se $b = 0$. A estrutura de restrições a ser utilizada é então

$$f \leq Mb \tag{5.4}$$

onde M é um inteiro positivo tal que $f \leq M$. Nota-se que este é um caso particular da estrutura

[1] a função indicatriz (ou indicadora), denotada por $\mathbb{1}(x \in A)$ é uma função binária que vale 1 se $x \in A$ e 0 senão, onde A é um conjunto qualquer. Esta função é utilizada para representar o pertencimento a um conjunto ou a verificação de uma condição.

"se - então" apresentada acima.

Queremos então expressar a seguinte condição: no caso do veículo k visitar o cliente j após o cliente i, e se o tempo já trabalhado por k em um determinado dia ao servir o cliente i somado ao tempo de trajeto até j ultrapassar a jornada máxima de trabalho então o tempo de início do atendimento do cliente j pelo veículo k deve levar em conta o tempo de descanso necessário para o veículo k.

Matematicamente temos: seja D o número de horas de um dia (24 horas), a_d o número máximo de horas trabalhadas por dia e t_d o tempo mínimo de descanso, se $(s_{ik}+u_i) \bmod D + t_{ij} \geq a_d$ então $s_{ik}+u_i+t_{ij}+\lfloor\frac{(s_{ik}+u_i) \bmod D+t_{ij}}{a_d}\rfloor t_d \leq s_{jk}, \forall i \in C \cup \{0\}, \forall j \in C \cup \{n+1\}, \forall k \in K$, e estas condições são condicionais a $x_{ijk} = 1$. Que escrevemos na forma de restrições

$$
\begin{aligned}
&(s_{ik}+u_i) \bmod D + t_{ij} - a_d + \varepsilon \\
&\qquad \leq M'(1-z_{ik}) + M''(1-x_{ijk}), \qquad \forall (i,j) \in E, \forall k \in K \qquad (5.5)\\
&(s_{ik}+u_i+t_{ij}+\lfloor\frac{(s_{ik}+u_i) \bmod D+t_{ij}}{a_d}\rfloor t_d - s_{jk}) \\
&\qquad \leq M' z_{ik} + M''(1-x_{ijk}), \qquad \forall (i,j) \in E, \forall k \in K \qquad (5.6)
\end{aligned}
$$

No entanto, as operações de módulo (mod)[2] e de divisão inteira ($\lfloor\cdot\rfloor$)[3] precisam ser linearizadas. Para a linearização da expressão $y = x \bmod a$ utilizamos então a seguinte formulação:

$$
\begin{aligned}
x &= ka + y && (5.7)\\
0 &\leq y \leq a - \varepsilon && (5.8)\\
k &\in \mathbb{N} && (5.9)
\end{aligned}
$$

Obtemos então o seguinte conjunto de restrições que modelam a inclusão do tempo de descanso nas rotas.

$$
\begin{aligned}
s_{ik} + u_i &= n^s_{ik} D + w^s_{ik}, && \forall (i,j) \in E, \forall k \in K && (5.10)\\
0 &\leq w^s_{ik} \leq D - \varepsilon, && \forall i \in C, \forall k \in K && (5.11)
\end{aligned}
$$

[2] a operação de módulo aqui é utilizada no sentido de resto de divisão inteira, comum em aritmética modular e em linguagens de programação.

[3] A operação de divisão inteira fornece a parte inteira da divisão, ignorando a parte fracionária. Por exemplo $\lfloor 3/2 \rfloor = 1$.

$$w^s_{ik} + t_{ij} - a_d + \varepsilon \leq M'(1 - z_{ik}) + M''(1 - x_{ijk}), \qquad \forall (i,j) \in E, \forall k \in K \tag{5.12}$$

$$w^s_{ik} + t_{ij} = n^t_{ijk} a_d + w^t_{ijk}, \qquad \forall (i,j) \in E, \forall k \in K \tag{5.13}$$

$$0 \leq w^t_{ijk} \leq a_d - \varepsilon, \qquad \forall (i,j) \in E \tag{5.14}$$

$$(s_{ik} + u_i + t_{ij} + n^t_{ijk} t_d - s_{jk}) \leq M' z_{ik} + M''(1 - x_{ijk}), \qquad \forall (i,j) \in E, \forall k \in K \tag{5.15}$$

$$n^s_{ik} \in \mathbb{N} \tag{5.16}$$

$$n^t_{ijk} \in \mathbb{N} \tag{5.17}$$

Uma hipótese para a validade dessas condições é que todos os veículos começam a jornada no mesmo horário com o referencial de horário $t = 0$ ("hora 0"), e então efetivamente a variável $w^s_{ik} = (s_{ik} + u_i) \bmod D$ representa o número de horas trabalhadas pelo veículo k em um determinado dia após servir o cliente i. A variável n^s_{ik} nos dá o número de dias que a viagem está durando até o momento e n^t_{ijk} é o número de períodos de descanso que será inserido (consideramos a possibilidade de inserir mais de um período de descanso, caso o tempo de viagem entre dois nós seja maior do que um dia).

Para tornar essa condição um pouco mais geral, podemos introduzir um parâmetro e_k que indica a hora de início do serviço do veículo k (ele deve começar todos os dias na mesma hora) e então $w^s_{ik} = (s_{ik} + u_i - e_k) \bmod D$ nos fornece a quantidade desejada. Suponha que $s_{ik} = 12$, $u_i = 1$ e $e_k = 8$, o que significa que o veículo começou a jornada às 8 horas da manhã, iniciou o atendimento ao cliente i ao meio dia e o serviço durou 1h. Temos então $(12 + 1 - 8) \bmod 24 = 5$ horas trabalhadas no dia após o atendimento de i. Em horizontes mais longos essa equação é válida também: suponha que $s_{ik} = 35$, $u_i = 1$ e $e_k = 8$, o que significa que o veículo começou a jornada à 8 da manhã, iniciou o atendimento ao cliente i às 11 da manhã do segundo dia e o serviço durou 1h. Temos então $(35 + 1 - 8) \bmod 24 = 4$ horas trabalhadas no dia. A hipótese de fixar os veículos em uma grade fixa de horários pode ser um pouco restritiva, então ao resolver o problema, podemos estabelecer uma variedade de valores para os diferentes e_k de maneira que a frota possa ter uma cobertura de horários maior.

A versão final das restrições é a seguinte:

$$s_{ik} + u_i - e_k = n^s_{ik} D + w^s_{ik}, \qquad \forall (i,j) \in E, \forall k \in K \tag{5.18}$$

$$0 \leq w^s_{ik} \leq D - \varepsilon, \qquad \forall i \in C, \forall k \in K \tag{5.19}$$

$$w^s_{ik} + t_{ij} - a_d + \varepsilon \leq M'(1 - z_{ik}) + M''(1 - x_{ijk}), \qquad \forall (i,j) \in E, \forall k \in K \tag{5.20}$$

$$w^s_{ik} + t_{ij} = n^t_{ijk} a_d + w^t_{ijk}, \qquad \forall (i,j) \in E, \forall k \in K \tag{5.21}$$

$$0 \leq w^t_{ijk} \leq a_d - \varepsilon, \qquad \forall (i,j) \in E \tag{5.22}$$

$$(s_{ik} + u_i + t_{ij} + n^t_{ijk} t_d - s_{jk}) \leq M' z_{ik} + M''(1 - x_{ijk}), \qquad \forall (i,j) \in E, \forall k \in K \tag{5.23}$$

$$n^s_{ik} \in \mathbb{N} \tag{5.24}$$

$$n^t_{ijk} \in \mathbb{N} \tag{5.25}$$

5.3 Resolução do problema em um exemplo reduzido

Descrevemos aqui uma versão simplificada do problema, com dados fictícios para ilustrar o modelo descrito acima e evidenciar o papel de algumas restrições.

5.3.1 Dados do exemplo

O exemplo é baseado em um método de geração de dados descrito no Apêndice A. Aqui utilizaremos um exemplo reduzido com 10 lojas a serem roteadas, baseada na instância de tipo C1 de Solomon (SOLOMON, 2005).

As coordenadas dos pontos e a matriz de distâncias correspondente são apresentadas nas Tabelas 9 e 10. As distâncias estão arredondadas para o inteiro mais próximo e possuem um fator multiplicativo de 10 km (ou seja, calcula-se a distância cartesiana e em seguida multiplica-se o valor por 10). Um diagrama com as localizações é também apresentado na Figura 12.

Tabela 9: Localizações do exemplo

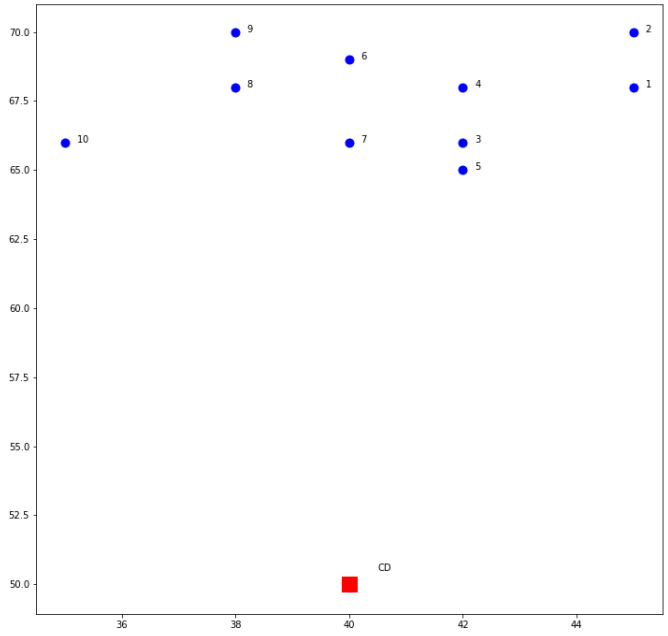
Nó	X	Y
CD	40	50
Loja 1	45	68
Loja 2	45	70
Loja 3	42	66
Loja 4	42	68
Loja 5	42	65
Loja 6	40	69
Loja 7	40	66
Loja 8	38	68
Loja 9	38	70
Loja 10	35	66

Fonte: (SOLOMON, 2005)

A frota é composta por $K = 3$ tipos de veículos, cujas características estão descritas na Tabela 10.

Por fim, as características de cada loja estão descritas na Tabela 12. Para simplificar, a

Figura 12: Posicionamento dos nós do exemplo



Fonte: elaborado pelo autor

demanda é apresentada já agregada em volume e valor e restrição de peso é ignorada. A janela de tempo é representada já incluindo cálculo do dia da entrega, como descrito no Capítulo 4. Consideramos também que todos os veículos são aceitos por todas as lojas.

5.3.2 Solução do exemplo

A solução para o exemplo descrito tem o custo de R$ 6793,30 e as rotas e tempos de serviço estão indicadas na Figura 13 e nas Tabelas 13 e 14. Nota-se que todas as janelas de tempo das lojas e o tempo de descanso dos veículos são respeitados, além das restrições de capacidade.

5.4 Detalhes sobre implementação

Para a implementação do método exato, utilizou-se da interface (API) em Python do *software IBM ILOG CPLEX Optimization Studio* versão 12.9 com suas configurações padrão.

A escolha da interface em Python e não da mais tradicional OPL se deve ao fato da linguagem Python ser mais flexível, permitindo uma maior liberdade no código, na geração de

Tabela 10: Distâncias do exemplo

	CD	1	2	3	4	5	6	7	8	9	10
CD	0										
1	187	0									
2	206	20	0								
3	161	36	50	0							
4	181	30	36	20	0						
5	151	42	58	10	30	0					
6	190	51	51	36	22	45	0				
7	160	54	64	20	28	22	30	0			
8	181	70	73	45	40	50	22	28	0		
9	201	73	70	57	45	64	22	45	20	0	
10	168	102	108	70	73	71	58	50	36	50	0

Fonte: elaborado pelo autor

Tabela 11: Dados da frota do exemplo

Tipo	Capacidade		Custo		Disponibilidade
	Volume	Valor	Fixo	Frete	
1	24	101.000,00	57,00	2,10	10
2	36	101.000,00	79,80	2,60	10
3	46,8	101.000,00	95,76	3,10	10

Fonte: elaborado pelo autor

exemplos e na leitura de dados. A linguagem Python vem ganhando popularidade ao longo dos anos, especialmente na comunidade científica, mas também em vários outros domínios (como na análise de dados, desenvolvimento *web*, etc.) como evidenciado na pesquisa realizada pela *Python Software Foundation* junto com a empresa *JetBrains* (PYTHON FOUNDATION, 2018).

Além disso, utilizou-se da ferramenta *Jupyter Notebook*, uma ferramenta *web* interativa e *open-source* que cria documentos contendo células de código, equações, visualizações, texto e imagens que podem ser rodadas independentemente, criando uma experiência mais interativa do código. Essa ferramenta tem se tornado bastante popular em vários domínios, principalmente para a análise de dados e pesquisa (PERKEL, 2018).

Os testes foram realizados utilizando-se um computador de sistema operacional *Microsoft Windows 10 Pro*, com processador Intel®Core™i7-4790 3.60 GHz, 3601 MHz, 4 núcleos e 8 processadores lógicos, e memória RAM instalada de 16,0 GB.

Os códigos utilizados na implementação do método exato e na implementação da heurística (sobre a qual se discorrerá no6) estão disponíveis no Apêndice B.

Tabela 12: Dados das lojas do exemplo

Loja	Janela (h)		Tempo de serviço (min)	Demanda	
	Início	Fim		Volume (m^3)	Valor (R$)
1	12	22	157	29,75	45.750,85
2	60	69	137	20,75	21.189,01
3	59	73	108	30,36	19.977,13
4	35	37	133	33,00	53.758,27
5	60	70	118	34,30	45.930,86
6	11	13	86	18,92	18.691,27
7	37	47	75	5,71	3.731,50
8	58	72	162	28,09	28.394,25
9	57	69	135	10,40	15.230,61
10	54	61	158	3,83	4.347,49

Fonte: elaborado pelo autor

Tabela 13: Rotas da solução ótima do exemplo

Indice do veículo	Tipo do veículo	Trajeto
11	2	CD, 5, CD
13	2	CD, 1, 7 CD
14	2	CD, 4, CD
19	2	CD, 3, CD
25	3	CD, 10, 9, 8 CD
28	3	CD, 6, 2, CD

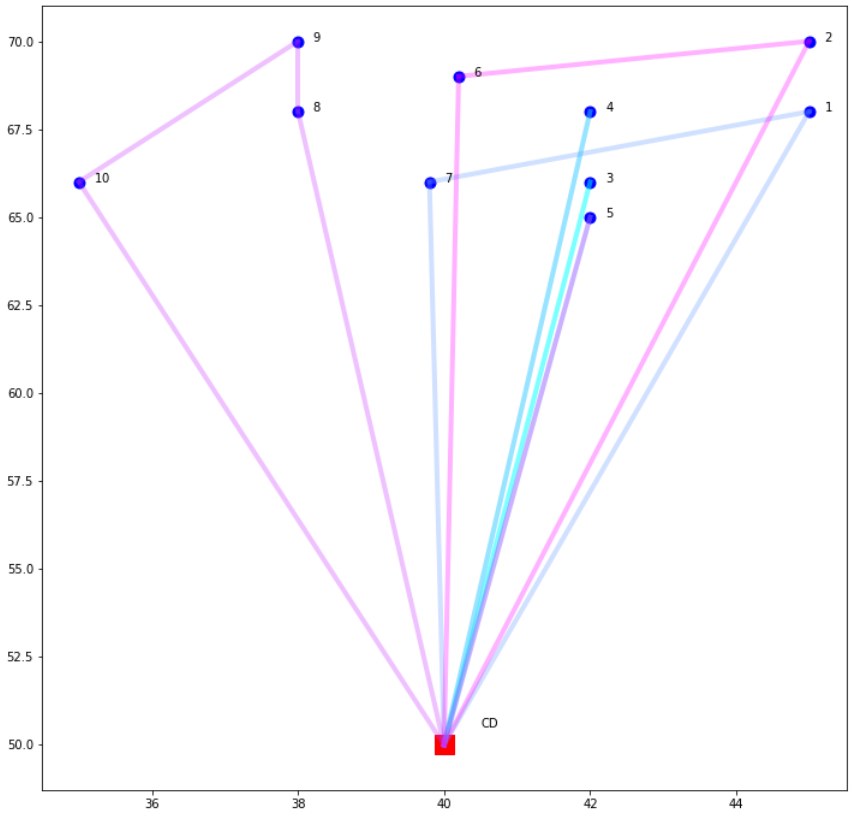
Fonte: elaborado pelo autor

5.5 Solução exata de instâncias de tamanho crescente

A utilização do método exato para resolução do problema só é factível para instâncias pequenas, ou seja, com número reduzido de clientes e veículos. Para compreender o comportamento do modelo conforme as instâncias crescem e evidenciar a necessidade do método heurístico, foram realizados experimentos com um número crescente de clientes, até atingir o limite do *hardware* utilizado.

As instâncias base utilizadas nesses experimentos são as instâncias de Solomon (SOLOMON, 1987), onde selecionamos a cada vez os n primeiros clientes para a construção do problema. O número de veículos disponíveis varia em função de n para garantir que o problema seja sempre factível. As demais características do problema (janelas de tempo, demandas e capacidades) são definidas segundo a metodologia descrita no Apêndice A. No Apêndice A

Figura 13: Representação da solução ótima do exemplo



Fonte: elaborado pelo autor

também se encontra uma descrição mais detalhada das classes de instâncias.

Os experimentos realizados, além de fornecer uma estimativa do tempo necessário para a resolução dos problemas, fornecem o valor exato das respectivas instâncias, valores que serão utilizados posteriormente para a validação do método heurístico.

Os resultados dos experimentos realizados nos problemas de tipo C1 e C2 podem ser vistos respectivamente nas Tabelas 15 e 16, onde são apresentados dados relativos ao tamanho do problema (número de clientes e de veículos), o tamanho resultante em termos de variáveis e restrições e finalmente o tempo de pré-processamento e de resolução, ambos em segundos. Incluímos o tempo de pré-processamento: mesmo que negligenciável no início, este torna-se bastante significativo quando o tamanho das instâncias aumenta. Excluímos do tempo de pré-processamento o tempo de leitura dos dados, então este mede apenas o tempo necessário para a criação das variáveis e restrições no *CPLEX*.

Podemos constatar com base nos resultados o crescimento do número de variáveis, restrições e também dos tempos de pré-processamento e de solução conforme o número de clientes aumenta. A taxa de crescimento evidencia o caráter **NP**-difícil do problema de roteirização. Para

Tabela 14: Tempos de início do serviço da solução ótima do exemplo

Loja	Veículo	Início do serviço (h)
1	13	12
2	28	60
3	19	59
4	14	35
5	11	60
6	28	11
7	13	37
8	25	59.75
9	25	57.25
10	25	54

Fonte: elaborado pelo autor

as duas classes de problema, atinge-se um limite de memória do computador com 20 e 25 clientes (classes C1 e C2 respectivamente) após um tempo de cálculo grande (28.261,78 segundos e 51.138,82 segundos respectivamente). A ordem de grandeza do tamanho desses problemas é inferior àquela encontrada na vida real, como visto no Capítulo 2 (na região Norte, que tem o menor número de lojas, a LASA possui 95 lojas), então imperativamente precisamos de um método heurístico para buscar uma solução para o problema real, o que será desenvolvido no próximo capítulo.

Tabela 15: Resolução exata de instâncias de tamanho crescente, classe C1

Clientes	Veículos	Variáveis	Restrições	Tempo c1	
				Pré-processamento (s)	Solução (s)
1	3	54	49	0,64	0,02
2	6	204	212	0,90	0,02
3	9	504	579	2,32	0,02
4	12	1.008	1.240	5,03	0,01
5	15	1.770	2.285	10,02	0,02
6	18	2.844	3.804	16,72	0,03
7	21	4.284	5.887	25,89	0,08
8	24	6.144	8.624	37,43	0,13
9	27	8.478	12.105	51,71	1,30
10	30	11.340	16.420	72,89	4,02
11	33	14.784	21.659	84,30	6,33
12	36	18.864	27.912	104,11	6,16
13	39	23.634	35.269	126,43	6,36
14	42	29.148	43.820	151,34	7,86
15	44	34.672	52.463	199,78	9,30
16	45	39.960	60.811	236,30	10,98
17	46	45.724	69.937	280,84	15,92
18	47	51.982	79.871	318,20	35,45
19	48	58.752	90.643	374,83	82,39
20	49	66.052	102.283	415.08	$*$[1]

[1] atingido o limite de memória do computador

Fonte: elaborado pelo autor

Tabela 16: Resolução exata de instâncias de tamanho crescente, classe C2

Clientes	Veículos	Variáveis	Restrições	Tempo c2	
				Pré-processamento (s)	Solução (s)
1	3	54	49	0,68	0,19
2	6	204	212	1,23	0,02
3	9	504	579	2,63	0,02
4	12	1.008	1.240	5,36	0,02
5	15	1.770	2.285	9,38	0,02
6	18	2.844	3.804	15,39	0,11
7	21	4.284	5.887	29,14	0,20
8	24	6.144	8.624	36,76	0,22
9	27	8.478	12.105	62,64	0,94
10	30	11.340	16.420	73,66	4,69
11	33	14.784	21.659	95,47	1,55
12	36	18.864	27.912	127,05	1,44
13	39	23.634	35.269	160,86	1,05
14	42	29.148	43.820	213,38	2,66
15	44	34.672	52.463	259,18	5,00
16	45	39.960	60.811	277,77	6,16
17	46	45.724	69.937	327,88	6,47
18	47	51.982	79.871	393,36	18,19
19	48	58.752	90.643	436,02	50,14
20	49	66.052	102.283	492,52	516,11
21	50	73.900	114.821	752,09	1217,13
22	51	82.314	128.287	683,23	455,97
23	52	91.312	142.711	629,29	3113,22
24	53	100.912	158.123	700,17	1364,72
25	54	111.132	174.553	791,56	$*$[1]

[1] atingido o limite de memória do computador

Fonte: elaborado pelo autor

6 HEURÍSTICAS

Este capítulo será dedicado à apresentação dos métodos heurísticos utilizados para resolver o problema alvo deste trabalho. Como descrito anteriormente no Capítulo 3, o problema de roteirização de veículos pertence à classe de problemas **NP**-difícil (***NP**-hard*) e uma das consequências desse fato se dá no crescimento exponencial do tempo de busca de soluções exatas quando a dimensão do problema aumenta, fato também evidenciado pelos testes computacionais realizados no Capítulo 5. Deste modo, a busca de soluções ótimas em situações reais como as encontradas nesse trabalho está muitas vezes fora do alcance de um ponto de vista técnico, seja por falta de capacidade computacional (memória e processamento) ou por tempo insuficiente.

Para lidar com esse problema, uma solução é o desenvolvimento e a utilização de métodos heurísticos para a busca de soluções que não são garantidamente ótimas, mas que podem aprimorar soluções existentes e que podem ser obtidas em um horizonte de tempo razoável para tamanhos de problemas variados.

O capítulo inicia com a apresentação da classe de heurísticas escolhida. Em seguida, são apresentadas as adaptações necessárias a serem feitas no método heurístico para que este possa ser aplicado no problema tratado neste trabalho, e posteriormente a aplicação da heurística no mesmo problema reduzido do Capítulo 5.

Na seção seguinte, mostramos o método testado para a calibragem dos coeficientes da heurística e os valores encontrados que serão adotados em seguida, onde realizamos experimentos comparando as soluções encontradas pelos métodos exato e heurístico para avaliar a qualidade das soluções propostas pela heurística.

6.1 Heurísticas de inserção

Escolhemos trabalhar com uma classe de métodos heurísticos construtivos. Nessas heurísticas as rotas são criadas sequencialmente até esgotar a capacidade de um veículo ou até que a adição

de uma loja extra não seja mais interessante segundo os critérios da heurística (a serem apresentados abaixo).

Esta classe de métodos heurísticos, também conhecida pelo nome de heurísticas de inserção, foi inicialmente desenvolvidas por (SOLOMON, 1987) e posteriormente estudada em outros trabalhos como em (DULLAERT *et al.*, 2002; KRITIKOS; IOANNOU, 2013; RONCONI; MANGUINO, 2016).

Definimos uma rota como sendo um veículo e uma lista ordenada de clientes atendidos por esse veículo. Procedemos da seguinte maneira: inicialmente, uma rota é inicializada com um veículo atendendo apenas a um cliente. Em seguida, os próximos clientes são inseridos à rota atual segundo dois critérios $C1$ e $C2$ até que a rota se torne inviável ou se o ganho marginal pela adição de um cliente adicional à rota seja negativo; então, uma nova rota é criada e o processo é repetido até que todos os clientes estejam roteados. O critério $C1$ avalia o impacto negativo ao adicionar a loja c à rota atual na posição de menor custo e identifica essa posição levando em conta os seguintes fatores (subcritérios):

- aumento de custo pela mudança de veículo,
- aumento do tempo total da rota,
- aumento da distância total da rota.

Já o critério $C2$ avalia o benefício de ter a loja c inserida na rota atual (na posição identificada pelo critério $C1$), comparado ao custo de criar uma nova rota exclusiva para c. A partir da maximização do critério $C2$ define-se o cliente a ser inserido na rota atual.

A escolha do primeiro cliente para a inicialização da rota pode seguir um dos critérios:

- Cliente mais distante ainda não atendido
- Cliente com janela de tempo mais urgente ainda não atendido
- Cliente não atendido com a menor ponderação de distância e duração do trajeto direto ao cliente

A descrição acima pode ser formulada matematicamente, e reproduzimos a seguir a formulação proposta em (DULLAERT *et al.*, 2002). Primeiro temos o cálculo de $C1$ e dos subcritérios que o compõem

$$c_1(i,u,j) = \min_p [c_1^d(i_{p-1},u,i_p)], \qquad p = 1,\ldots,m \tag{6.1}$$

com

$$c_1^d(i,u,j) = \sum_{i=1}^{3} \alpha_{1i} c_{1i}(i,u,j) \tag{6.2}$$

$$c_{11}(i,u,j) = d_{iu} + d_{uj} - \mu d_{ij} \tag{6.3}$$

$$c_{12}(i,u,j) = s_j^{novo} - s_j \tag{6.4}$$

$$c_{13}(i,u,j) = cf(Q^{\text{novo}}) - cf(Q) \tag{6.5}$$

e em seguida o cálculo de $C2$:

$$c_2(i,u^*,j) = \max_u [c_2^d(i,u,j)] \tag{6.6}$$

considerando

$$c_2^d(i,u,j) = \lambda(d_{0u} + t_{0u}) + s_u + cf(q_u) - c_1(i,u,j) \tag{6.7}$$

Os argumentos (i,u,j) das funções acima representam o calculo do critério em questão para a inserção do cliente u entre as posições i e j na rota. O calculo de c_1 na equação (6.1) envolve a busca em todas as posições p da rota atual (de tamanho total m) para a inserção da loja u, buscando a posição de menor custo. Temos segundo a equação (6.2) que o custo c_1 é uma ponderação (com pesos α_{11}, α_{12} e α_{13}) das funções c_{11}, c_{12} e c_{13}, que são os subcritérios que representam respectivamente o aumento da distância da rota, o aumento do tempo da rota e o aumento nos custos fixos da rota. O aumento do custo fixo representado na equação de c_{13} funciona da seguinte maneira: Q representa o volume atualmente transportado na rota, Q^{novo} representa o volume após a adição do novo cliente, e cf é uma função que fornece o custo fixo do menor veículo cuja capacidade é maior ou igual a Q ou Q^{novo}.

De maneira equivalente, o cálculo de c_2 na equação (6.6) busca dentre todos os clientes possíveis u aquele que maximiza o valor da equação (6.7), que por sua vez tem suas similaridades com o cálculo de c_1. Temos os termos na seguinte ordem: a distância, o tempo, o tempo de início do serviço e o custo do veículo, todos calculados para um atendimento exclusivo do cliente u, subtraídos do custo c_1 de inserção de u na rota. O termo q_u é o volume da demanda do cliente u. A equação (6.7) representa então efetivamente o que foi descrito anteriormente, comparando o custo total de uma rota exclusiva para u contra a alternativa de inserir u na rota atual.

Os pesos α, λ e μ são hiper-parâmetros do método (calibrados separadamente) necessários para balancear os diferentes valores dos custos e suas unidades (tempos, distâncias e unidades monetárias).

Uma característica interessante da heurística apresentada está na sua flexibilidade. Os critérios $C1$ e $C2$ podem ser refinados e adaptados para o problema em questão, incluindo por exemplo uma estrutura diferente de custos (MANGUINO; RONCONI, 2012). Em (DULLAERT *et al.*, 2002), por exemplo, a função c_{13} pode tomar três formas diferentes, ACS, AOOS e AROS, que descrevemos abaixo e cujas equações são indicadas na Tabela 17.

- O critério *Adapted Combined Savings* (ACS) mede a diferença entre o custo fixo do veículo capaz de transportar a carga da rota antes e depois da inserção do novo cliente ("rota atualizada")
- O critério *Adapted Optimistic Opportunity Savings* (AOOS) subtrai do ACS o custo fixo do menor veículo capaz de atender à demanda não utilizada da rota atualizada. Aqui o termo $\bar{Q}$ representa a capacidade máxima do veículo em questão.
- E o *Adapted Realistic Opportunity Savings* (AROS) subtrai do ACS o custo fixo do maior veículo com capacidade menor ou igual à capacidade em excesso da rota atualizada, denotada cf' (note a diferença com a função cf que retorna uma quantidade diferente). Uma função binária $\delta(\omega)$ controla essa subtração, que é efetiva somente se um veículo maior é necessário para atender a rota atualizada.

Tabela 17: Critérios da heurística apresentada por (DULLAERT *et al.*, 2002)

Critério	Equação
ACS	$cf(Q^{novo}) - cf(Q)$
AOOS	ACS $-cf(\bar{Q}^{novo} - Q^{novo})$
AROS	ACS $-\delta(\omega)cf'(\bar{Q}^{novo} - Q^{novo})$

Fonte: (DULLAERT *et al.*, 2002)

6.2 Adaptações da heurística para o problema em questão

A heurística propostas em (DULLAERT *et al.*, 2002) trata do problema de roteirização de veículos com frota heterogênea e janelas de tempo. Como o problema apresentado nesse trabalho possui mais especificidades (restrição dupla de capacidade, tempo de descanso, *site dependence*), o procedimento descrito acima não pode ser aplicado diretamente e precisa de algumas adaptações para garantir que todas as restrições sejam respeitadas e uma solução coerente seja produzida.

O principal fator a se levar em conta é a relação dos clientes e dos veículos que podem atendê-los, pois isso influencia diretamente no algoritmo da heurística. Como cada cliente possui um conjunto de tipos de veículos que pode atendê-los, então ao comparar candidatos para a inserção, se um cliente não pode ser atendido pelo veículo atual da rota (por ser incompatível), ele não será considerado candidato para ser inserido, a não ser que uma mudança de veículo seja factível.

A viabilidade na mudança de veículo também precisa ser verificada durante o cálculo de c_1. Consideramos que a troca do veículo é possível somente se o novo veículo é aceito por todos os clientes existentes da atual rota, senão a troca é infactível.

No cálculo de c_{13} vamos considerar o critério ACS, por simplicidade e também pelos resultados em (DULLAERT *et al.*, 2002) que mostram que as diferenças entre os 3 critérios não são muito grandes. Uma modificação ao critério ACS será considerada: junto com o acréscimo do custo fixo, adicionaremos o custo adicional de frete multiplicado pela distância total da rota, já que a mudança no tipo de veículo traz consigo também uma mudança no valor do frete por quilômetro. Assim a equação se torna

$$c_{13}(i,u,j) = cf(Q^{\text{novo}}) - cf(Q) + (cv^{\text{novo}} - cv) \sum_{(m,n)\in \text{rota}\cup\{u\}} d_{mn} \tag{6.8}$$

Outro aspecto importante do problema tratado é a questão do tempo máximo de trabalho e tempo mínimo de descanso dos motoristas. Esse aspecto se integra mais facilmente na heurística, pois basta verificar a o tempo de trabalho a cada inserção na rota e adicionar os respectivos tempos de descanso conforme necessário. Temos as mesmas condições que aquelas apresentadas no modelo de otimização, mas sem a necessidade de linearização das restrições. Podemos expressar diretamente a condição: para uma determinada rota cujo veículo é k, se o cliente j é atendido após o cliente i e se $(s_{ik}+u_i) \bmod D + t_{ij} \geq a_d$ então

$$s_{ik} + u_i + t_{ij} + \lfloor \frac{(s_{ik}+u_i) \bmod D + t_{ij}}{a_d} \rfloor t_d \leq s_{jk}, \forall i \in C \cup \{0\}, \forall j \in C \cup \{n+1\}, \forall k \in K.$$

Assim, temos que

$$s_{jk} = \max(a_j, s_{ik} + u_i + t_{ij} + \lfloor \frac{(s_{ik}+u_i) \bmod D + t_{ij}}{a_d} \rfloor t_d),$$

sujeito a $s_{jk} \in [a_j, b_j]$ (respeitar a janela de tempo). É interessante notar que o aumento no início do tempo de serviço do próximo cliente já é penalizado pelo critério $c1$ com o termo c_{12}; caso um descanso seja necessário, a penalidade aumenta ainda mais.

Para permitir uma flexibilidade maior no balanceamento dos critérios, serão incluídos pesos

β_i, similares aos parâmetros α, no cálculo de c_2. Esses hiper-parâmetros serão calibrados no capítulo seguinte. Temos então para o cálculo de $C1$

$$c_1(i,u,j) = \min_p [c_1^h(i_{p-1},u,i_p)], \qquad p = 1,\ldots,m \tag{6.9}$$

com

$$c_1^h(i,u,j) = \sum_{k=1}^{3} \alpha_k c_{1k}(i,u,j) \tag{6.10}$$

$$c_{11}(i,u,j) = d_{iu} + d_{uj} - \mu d_{ij} \tag{6.11}$$

$$c_{12}(i,u,j) = s_j^{novo} - s_j \tag{6.12}$$

$$c_{13}(i,u,j) = cf(Q^{\text{novo}}) - cf(Q) + (cv^{\text{novo}} - cv) \sum_{(m,n)\in \text{rota}\cup\{u\}} d_{mn} \tag{6.13}$$

e para o cálculo de $C2$:

$$c_2(i,u^*,j) = \max_u [c_2^h(i,u,j)] \tag{6.14}$$

considerando

$$c_2^h(i,u,j) = \left[\sum_{k=1}^{3} \beta_k c_{2k}(i,u,j)\right] - c_1(i,u,j) \tag{6.15}$$

$$c_{21}(i,u,j) = 2d_{0u} \tag{6.16}$$

$$c_{22}(i,u,j) = s_u + t_{0u} \tag{6.17}$$

$$c_{23}(i,u,j) = cf(q_u) + cv \cdot 2d_{0u} \tag{6.18}$$

O procedimento geral da heurística está descrito no algoritmo 1 e algumas rotinas auxiliares

estão descritas nos algoritmos 2, 3, 4 e 5.

Entrada: Localizações, distâncias, custos, frota
Saída: Rota viável para o problema
início
  marcar todo cliente c como não roteado;
  enquanto $\exists$ *c não roteado* **faça**
    $c_{max} =$ cliente mais distante não roteado;
    $rota =$ **criar_rota**(c_{max});
    $roteado[c_{max}] = True$;
    $candidatos =$ **identificar_candidatos**$(rota)$;
    enquanto $candidatos \neq \emptyset$ **faça**
      para $c \in candidatos$ **faça**
        $C1(c) =$ **calcular_c1**$(c, rota)$;
        $C2(c) =$ **calcular_c2**$(c, rota)$;
      fim
      se $\exists\, C2(c) > 0$ **então**
        $c_{inserir} = \arg\max_c C2(c)$;
        Adicionar $c_{inserir}$ à rota;
        $roteado[c_{inserir}] = True$;
        $candidatos =$ **identificar_candidatos**$(rota)$;
      fim
    fim
  fim
fim

Algoritmo 1: Heurística de inserção

Entrada: Uma loja c, lista de veículos disponíveis K
Saída: Uma rota
início
  Selecionar veículo $k \in K$ de menor custo tal que $demanda(c) \leq capacidade(k)$
  retorna $\boldsymbol{rota(capacidade = K, elementos = [c])}$
fim

Algoritmo 2: Função **criar_rota**

Entrada: Uma rota, lista de clientes não roteados, lista de veículos disponíveis K
Saída: Uma lista de clientes candidatos à inserção na rota
início
  Capacidade disponível =
   Capacidade(maior veiculo disponível compatível com todos os clientes da rota) −
   demanda_total($rota$);
  $candidatos = \{c \in C$:
   c não roteado, compatível com o veículo atual da rota e $demanda(c) <$
   Capacidade disponível$\}$;
  retorna $candidatos$
fim

Algoritmo 3: Função **identificar_candidatos**

Entrada: Uma loja c, uma rota, distâncias, tempos, custos fixos
Saída: O custo $C1$ de inserção de c na rota
início
  retorna *Custo* $C1$ *segundo a equação dada em* (6.10)
fim

Algoritmo 4: Função **calcular_c1**

6.3 Teste do método heurístico

Para validar a heurística apresentada, utilizaremos o exemplo apresentado anteriormente no Capítulo 5, de modelagem do problema, comparando os resultados da heurística à solução exata obtida nesse exemplo. No próximo capítulo apresentaremos também uma bateria de testes mais extensa para comparação e validação do método.

A solução apresentada pela heurística foi de R$ 7479.84, em torno de 10% acima do ótimo, sendo que a solução exata foi de R$ 6793,30. As rotas obtidas são representadas na Figura 14 e na Tabela 18, os tempos de serviço são apresentados na Tabela 19. Nota-se que a distância média de rotas é bem similar entre as duas soluções: 375 km no caso exato e 377 km no caso heurístico. No entanto, a solução heurística fornece 7 rotas comparado a 6 rotas na solução ótima, o que acaba impactando o custo final da solução. Isso acaba ilustrando uma característica de certas classes de heurísticas, que por realizarem uma busca local otimizando de maneira gulosa (*greedy*)[1] um critério alternativo (neste caso os critérios $c1$ e $c2$) acabam atingindo um ótimo local sem no entanto atingir o ótimo global (dado pela solução exata).

[1]Segundo (CORMEN *et al.*, 2009), um algoritmo guloso (*greedy*) é um tipo de algoritmo de otimização que sempre escolhe a opção que parece a melhor em um determinado momento. Isto é, ele escolhe uma opção localmente ótima, esperando assim atingir a solução globalmente ótima. É uma técnica de *design* de algoritmos bastante utilizada quando o problema a ser resolvido pertence à classe **NP**-difícil, pois assim pode se obter uma solução aproximada em tempo polinomial.

Entrada: Uma rota, uma lojas candidata c, o seu custo $C1$, distâncias, tempos, custos fixos
Saída: O custo $C2$ de inserção de c na rota
início
| **retorna** *Custo* $C2$ *segundo a equação dada em* (6.15)
fim

Algoritmo 5: Função **calcular_c2**

Tabela 18: Rotas da solução heurística do exemplo

Índice do veículo	Tipo do veículo	Trajeto
20	3	CD, 7, 10, 9, 2 CD
0	1	CD, 6, CD
10	2	CD, 1, CD
11	2	CD, 4, CD
12	2	CD, 8, CD
13	2	CD, 3, CD
14	2	CD, 5, CD

Fonte: elaborado pelo autor

6.4 Calibragem dos parâmetros do método heurístico

Como descrito anteriormente neste capítulo, as equações dos critérios da heurística de inserção envolvem os parâmetros α_k e β_k, $k = 1,2,3$, que precisam ser determinados para a utilização da heurística.

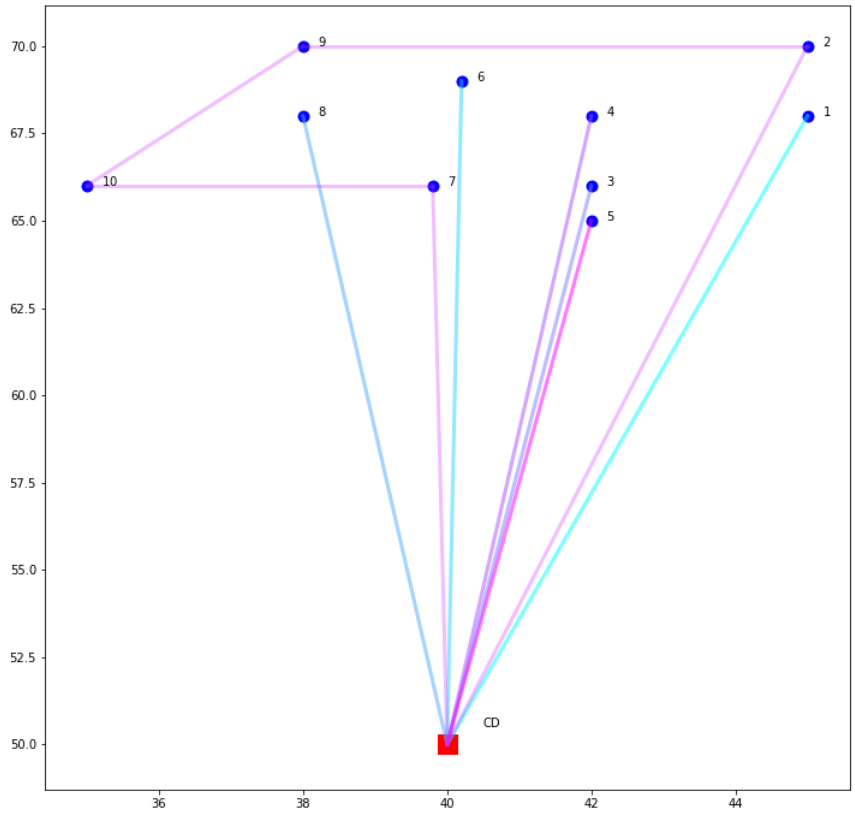
Esses parâmetros podem ser otimizados para os dados específicos do problema deste trabalho, o que não garante, por exemplo, a robustez dessa escolha de parâmetros quando houver uma mudança nos dados, hipóteses em que o resultado pode ser pior do que o esperado (fenômeno conhecido como *overfitting*). Para tentar evitar esse problema (comum em estatística, para a seleção de modelos), uma solução é calibrar os coeficientes em um conjunto de dados independente do conjunto de dados de teste.

Para isso, definimos uma instância referência para a calibragem e um conjunto de parâmetros a ser testado.

A técnica para determinar os parâmetros será uma busca em grelha - *Grid Search* (GS) (BERGSTRA; BENGIO, 2012), onde o conjunto de parâmetros a ser testado é uma grelha de valores discretos. Por exemplo, supondo que os valores de busca serão iguais para os 6 parâmetros, temos

$$\alpha_1 = v_1, v_2, \ldots, v_p$$

Figura 14: Representação da solução heurística do exemplo



Fonte: elaborado pelo autor

$$\alpha_2 = v_1, v_2, \ldots, v_p$$

$$\alpha_3 = v_1, v_2, \ldots, v_p$$

$$\beta_1 = v_1, v_2, \ldots, v_p$$

$$\beta_2 = v_1, v_2, \ldots, v_p$$

$$\beta_3 = v_1, v_2, \ldots, v_p$$

resultando assim em p^6 combinações diferentes a serem testadas. A escolha do número de valores p a serem testados envolve um *tradeoff* entre precisão e tempo. Quanto mais valores forem testados, maior será a chance de encontrar um bom resultado; no entanto, o tempo de cálculo (para testar todas as possibilidades) cresce exponencialmente. Para $p = 3$, por exemplo, são 729 combinações e para $p = 4$ temos 4096 combinações. A vantagem desse método de busca está na facilidade de sua implementação e eventual paralelização.

A escolha da instância teste também apresenta um *tradeoff* similar. Quanto maior a instância, maior o tempo necessário para encontrar uma solução; no entanto, uma instância muito pequena

Tabela 19: Tempos de início do serviço da solução heurística do exemplo

Loja	Veículo	Início do serviço (h)
1	10	12
2	20	60.38
3	13	59
4	11	35
5	14	60
6	0	11
7	20	37
8	12	58
9	20	57.25
10	20	54

Fonte: elaborado pelo autor

pode não ser representativa o suficiente para que os parâmetros encontrados generalizem corretamente para outros casos. Então é necessário encontrar um equilíbrio aceitável para o tamanho da instância e do número de valores a serem testados.

Escolhemos uma instância com 25 lojas, gerada da mesma maneira que o exemplo descrito no Capítulo 5. Para uma instância desse tamanho, o tempo médio de resolução pela heurística dura em média 28 segundos, então um teste com $p = 3$ valores duraria em torno de 5.7 horas e com $p = 4$ valores em torno de 32 horas.

Escolhemos o valor $p = 3$ para a realização de testes, pois também seria necessário repetir os testes em instâncias diferentes de mesmo tamanho, variando outras características como janelas de tempo, demandas e tamanho da frota. Assim, os valores testados foram todas as combinações entre 0, 0.5 e 1 para todos os parâmetros α_k e β_k, $k = 1,2,3$.

Os resultados dos testes não permitem a determinação de uma calibragem ótima única, pois a heurística acaba fornecendo soluções idênticas para diferentes combinações dos parâmetros. Assim, escolheu-se utilizar a média dos parâmetros ótimos obtidos. Para aumentar a gama de testes possíveis com os dados reais, utilizaremos também combinações de parâmetros encontradas em (DULLAERT *et al.*, 2002) (D1 e D2) e em (RONCONI; MANGUINO, 2016) (RM1 e RM2). As diferentes combinações que serão utilizadas são apresentadas na Tabela 20. Destaca-se que nos testes realizados com instâncias reduzidas, os diferentes conjuntos de parâmetros obtiveram resultados muito similares.

Tabela 20: Parâmetros para a heurística

Nome	α_1	α_2	α_3	β_1	β_2	β_3
D1	1.0	1.0	1.0	1.0	1.0	1.0
D2	1.0	0.0	1.0	1.0	0.0	1.0
RM1	0.9	0.0	0.3	0.2	0.0	1.0
RM2	0.2	0.0	0.8	0.0	0.1	0.2
GS	0.5	0.7	0.5	0.5	0.6	0.5

Fonte: (DULLAERT *et al.*, 2002; RONCONI; MANGUINO, 2016) e o autor

6.5 Comparação entre soluções exatas e o método heurístico

Com os conjuntos de parâmetros determinados (a partir da literatura e da calibragem), realizamos mais uma série de experimentos com as mesmas instâncias da Seção 5.5. Deste modo, buscou-se compreender o comportamento da heurística em função do tamanho crescente dos problemas resolvidos e comparar os resultados obtidos com aqueles originados da resolução do modelo exato. Os resultados utilizando o conjunto de parâmetros RM1 são apresentados nas Tabelas 21 e 22, respectivamente para as instâncias de classe C1 e C2, que são bastante semelhantes, diferindo na localização de certos clientes. A classe C2 possui clientes um pouco mais distantes nas instâncias testadas, com uma distância média dos arcos 27% superior à classe C1.

De acordo com os resultados, o aumento médio do custo da solução encontrada pela heurística com relação ao custo ótimo é de 2.6% e o maior aumento é de 10,19% nas instâncias da classe C1 e de 6.89% nas instâncias de classe C2. A Tabela 23 mostra outras estatísticas como o mínimo, máximo, mediana e desvio padrão. Segundo (VIDAL *et al.*, 2013), heurísticas construtivas produzem em geral soluções que estão entre 10% e 15% da solução ótima. Assim, a partir dos dados obtidos podemos afirmar que a heurística proposta produz soluções de boa qualidade para a classe de problemas que está sendo tratada. O próximo capítulo apresentará a aplicação da heurística nos dados reais da empresa.

Tabela 21: Resolução heurística de instâncias de tamanho crescente, classe C1

Clientes	Veículos	Tempo de solução (s)	Aumento no custo
1	3	0,05	0,00%
2	6	0,12	0,00%
3	9	0,25	0,00%
4	12	0,53	0,00%
5	15	0,75	0,00%
6	18	1,15	0,00%
7	21	1,83	1,17%
8	24	2,11	1,80%
9	27	3,18	9,36%
10	30	4,35	10,11%
11	33	6,68	10,19%
12	36	5,90	6,13%
13	39	7,02	2,64%
14	42	8,60	5,20%
15	44	10,39	4,46%
16	45	13,73	1,64%
17	46	16,13	1,52%
18	47	16,42	3,72%
19	48	17,71	1,69%

Fonte: elaborado pelo autor

Tabela 22: Resolução heurística de instâncias de tamanho crescente, classe C2

Clientes	Veículos	Tempo de solução	Aumento no custo
1	3	0,04	0,00%
2	6	0,15	0,00%
3	9	0,27	0,00%
4	12	0,56	0,00%
5	15	1,11	0,00%
6	18	1,39	0,00%
7	21	2,31	2,45%
8	24	2,94	2,24%
9	27	5,60	3,26%
10	30	5,05	4,07%
11	33	7,50	5,58%
12	36	9,78	3,95%
13	39	10,08	1,36%
14	42	13,09	0,51%
15	44	15,41	0,44%
16	45	16,23	0,46%
17	46	20,74	0,43%
18	47	23,16	0,36%
19	48	27,49	2,45%
20	49	32,88	3,54%
21	50	67,61	3,86%
22	51	36,03	3,73%
23	52	29,83	5,22%
24	53	33,06	6,89%

Fonte: elaborado pelo autor

Tabela 23: Estatísticas do aumento de custo com o método heurístico

Mínimo	1Q	Mediana	3Q	Máximo	Média	Desvio Padrão
0%	0%	1,69%	3,9%	10,19%	2,57%	2,84%

Fonte: elaborado pelo autor

7 ANÁLISE DOS RESULTADOS

Este capítulo é dedicado à apresentação e discussão da solução encontrada para o problema tratado no presente trabalho. Inicia-se apresentando a solução encontrada com base no método heurístico descrito no Capítulo 6 junto com uma comparação com o que é realizado atualmente pela empresa, evidenciando as economias obtidas. Apresenta-se em seguida um conjunto de análises de sensibilidade para avaliar a robustez do método apresentado contra mudanças no cenário do problema, como o aumento dos custos de transporte e variação das janelas de tempo.

7.1 Solução encontrada

A partir da heurística de inserção descrita no Capítulo 6 aplicada à instância real da empresa, encontrou-se uma solução de boa qualidade para o problema tratado neste trabalho, que aprimora os roteiros empregados atualmente pela LET'S. Temos 107 lojas a serem atendidas em um horizonte de 12 dias por um único CD, como descrito na Tabela 8. A solução proposta possui 17 rotas independentes e reduz o custo em 12% com relação ao roteiro atual empregado pela empresa.

A performance da solução proposta em outros indicadores, comparada ao roteiro atual, é apresentada na Tabela 24. Destaca-se a redução do número total de rotas, com o aumento do número médio de clientes por rota e da carga carregada pelos veículos, medida pelo fator de carga: razão entre o volume utilizado e o volume disponível do veiculo. Esse resultado está de acordo com o *design* da heurística, que busca inserir o máximo de clientes por rota até a sua saturação.

As rotas propostas podem ser vistas na Tabela 30 do Apêndice C.

Para a obtenção da solução final, foram realizados testes com diferentes coeficientes do método heurístico e também com a adição de uma etapa de normalização dos dados de entrada. Os conjuntos de coeficientes testados, descritos na Tabela 20 do Capítulo 6, foram baseados em resultados obtidos na literatura e também obtidos através de uma calibragem. A melhor solução

Tabela 24: Comparativo do roteiro otimizado com o roteiro atual

Indicador	Atual	Proposto
Número de rotas	29	15
Distância média por rota (km)	1.113,92	2.159,12
Distância máxima (km)	4.086,91	5.383,74
Número médio de clientes por rota	3.76	7.13
Número máximo de clientes por rota	10	14
Fator de carga médio (volume)	51,1%	96.6%
Fator de carga médio (valor)	26,5%	49.5%

Fonte: B2W e o autor

Tabela 25: Comparativo da utilização de veículos

Tipo de veículo	Atual	Proposto
Truck	22	13
3/4	4	1
Carreta	3	1

Fonte: B2W e o autor

foi obtida sem a normalização dos dados e o conjunto de parâmetros GS. A diferença entre o custo obtido com os parâmetros GS e o segundo melhor conjunto de parâmetros SM1 foi de 1.3%.

7.2 Análises de sensibilidade

Após a obtenção da solução a partir do método heurístico, buscou-se entender o comportamento da solução sob cenários alternativos em que os dados de entrada do problema sofrem algum tipo de mudança, como por exemplo um possível aumento nos custos de transporte devido a fatores externos. Este tipo de análise é conhecido como análise de sensibilidade.

Escolhemos analisar os seguintes cenários que poderiam alterar a solução obtida:

1. Aumento do custo de transporte
2. Aumento da demanda
3. Aumento das janelas de tempo
4. Variação no tempo de trabalho dos motoristas

7.2.1 Aumento do custo de transporte

O custo de transporte que é pago pela empresa é determinado pela transportadora Direct (comprada pela B2W em 2014). Como descrito no Capítulo 4, os custos são estimados pela empresa com base na disponibilidade habitual de caminhões e motoristas. É possível imaginar cenários em que os custos de transporte aumentam por razões externas, como alta demanda por veículos ou uma alta dos preços de combustível.

Supomos então uma série de cenários em que os custos de frete sofrem uma variação de -10%, +25%, +50%, +75%, +100%, tanto no custo fixo tanto no custo variável. O impacto de cada cenário no custo final da solução obtida pode ser visto na Tabela 26.

Tabela 26: Impacto do aumento dos custos de transporte no custo final da solução

Δ Custo fixo	Δ Custo variável					
	-10%	0%	25%	50%	75%	100%
-10%	-11.12%	-0.29%	18.45%	46.97%	71.03%	92.08%
0%	-10.83%	0.00%	18.73%	47.26%	71.32%	92.37%
25%	-10.10%	0.72%	19.44%	47.99%	72.05%	93.10%
50%	-9.38%	1.45%	20.14%	48.72%	72.77%	93.82%
75%	-8.65%	2.17%	20.84%	49.44%	73.50%	94.55%
100%	-7.93%	2.90%	21.55%	50.17%	74.23%	95.27%

Fonte: elaborado pelo autor

Chamamos atenção para alguns aspectos dos resultados obtidos:

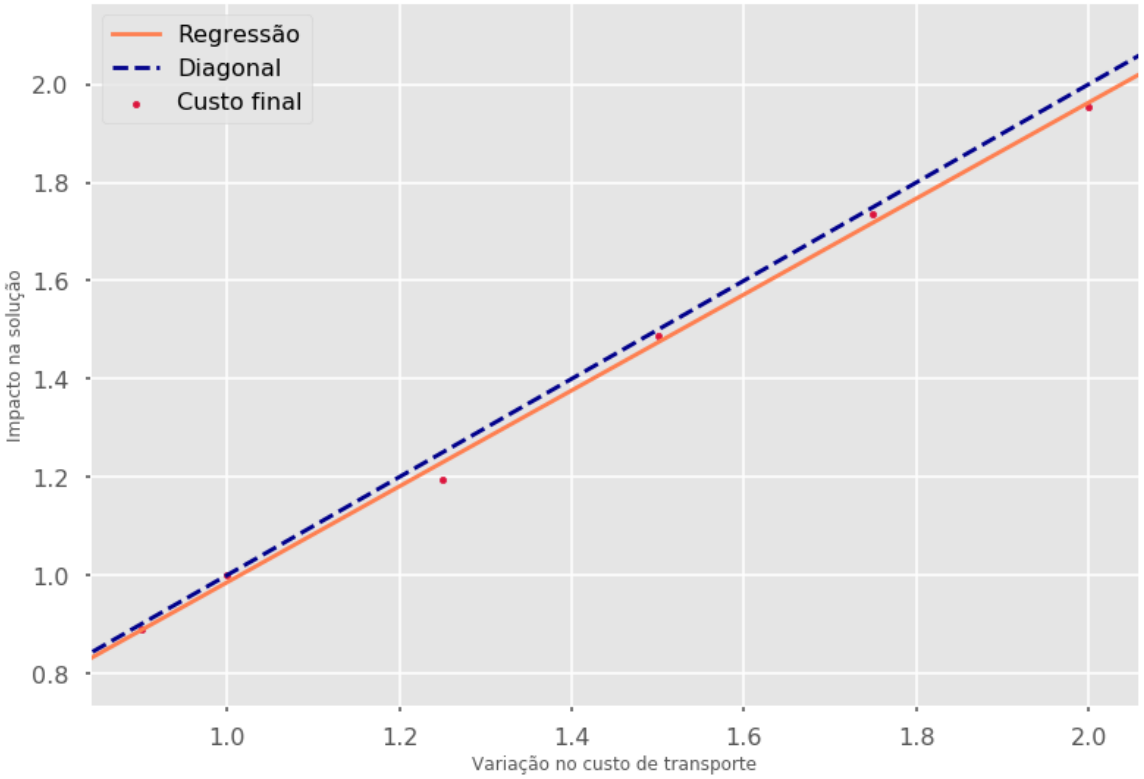
1. O custo variável possui um peso muito maior no custo final do que o custo fixo.
2. O custo final não cresce de maneira proporcional ao crescimento dos custos de transporte.

De fato, o primeiro ponto pode ser constatado pelo seguinte fato: a razão média entre o frete por quilometro e o custo fixo de um veículo é de aproximadamente 0.02, o que implica que os custos variáveis de um veículo superam os custos fixos a partir de 50 quilômetros rodados, então, dada a ordem de grandeza das distâncias encontradas no problema (a distância média de uma rota supera 1000 km), é esperado que os custos variáveis ocupem uma parcela maior do custo total.

O segundo ponto é constatado na diagonal principal da Tabela 26. Intuitivamente, um aumento idêntico dos custos fixos e variáveis se traduziria em um aumento de mesma proporção no custo final total, caso as rotas se mantivessem iguais. No entanto, este não é o comportamento que é observado nos dados: por exemplo, um aumento de 25% em ambos os custos se traduz em

um aumento de 19,44% do custo total. Isso se dá pelo fato de que o método heurístico pode se adaptar, propondo soluções diferentes, já que o aumento dos custos altera o valor dos critérios c_1 e c_2. Este comportamento pode ser visto na Figura 15, onde mostra-se o aumento no custo final em função da variação nos custos de transporte (uma regressão de equação $y = 0.0081 + 0.9773x$). Para efeitos de comparação, coloca-se a diagonal (a reta $y = x$) que representaria o caso se o aumento no custo final fosse proporcional.

Figura 15: Impacto da variação dos custos de transporte no custo final



Fonte: elaborado pelo autor

7.2.2 Variação da demanda

É realista considerar que a demanda das lojas pode variar ao longo do ano, podendo ser maior em períodos de festas ou datas comemorativas, por exemplo. Assim a matriz da LASA pode decidir aumentar a quantidade enviada às lojas para suprir essa demanda. De maneira similar, em períodos de menor movimento, podemos supor que as lojas recebem menos produtos em uma determinada entrega.

Consideramos então cenários em que o volume das entregas sofre variações de -10%, -5%, +5%, +10% e +25% e buscamos entender como as rotas e o custo final da solução se alteram com essa variação. Como uma boa parte das rotas já apresentava saturação de volume, espera-

se que o aumento da demanda provoque mudanças significativas nas rotas. Os resultados são apresentados na Tabela 27 e também no gráfico da Figura 16 onde mostra-se também a diagonal $y = x$ para efeito de comparação. Do mesmo modo que na análise anterior, nota-se que o custo não varia na mesma proporção que a variação na demanda, revelando que o método heurístico pode também adaptar as rotas nesse cenário para absorver melhor a variação da demanda.

Tabela 27: Impacto da variação da demanda na solução

Variação na demanda	Variação no custo
-10%	-20%
-5%	-2,8%
0% (referência)	0%
5%	0,1%
10%	3,2%
25%	14,8%

Fonte: elaborado pelo autor

7.2.3 Aumento das janelas de tempo

Decidiu-se verificar se o aumento das janelas de tempo tem algum impacto na solução obtida. Testou-se o aumento do final das janelas em 1 hora e 2 horas, e os resultados são apresentados na Tabela 28. É interessante ver que o aumento da duração das janelas eleva o custo da solução final. No entanto, a variação observada é pequena em valor absoluto, então é difícil afirmar se ela é significativa.

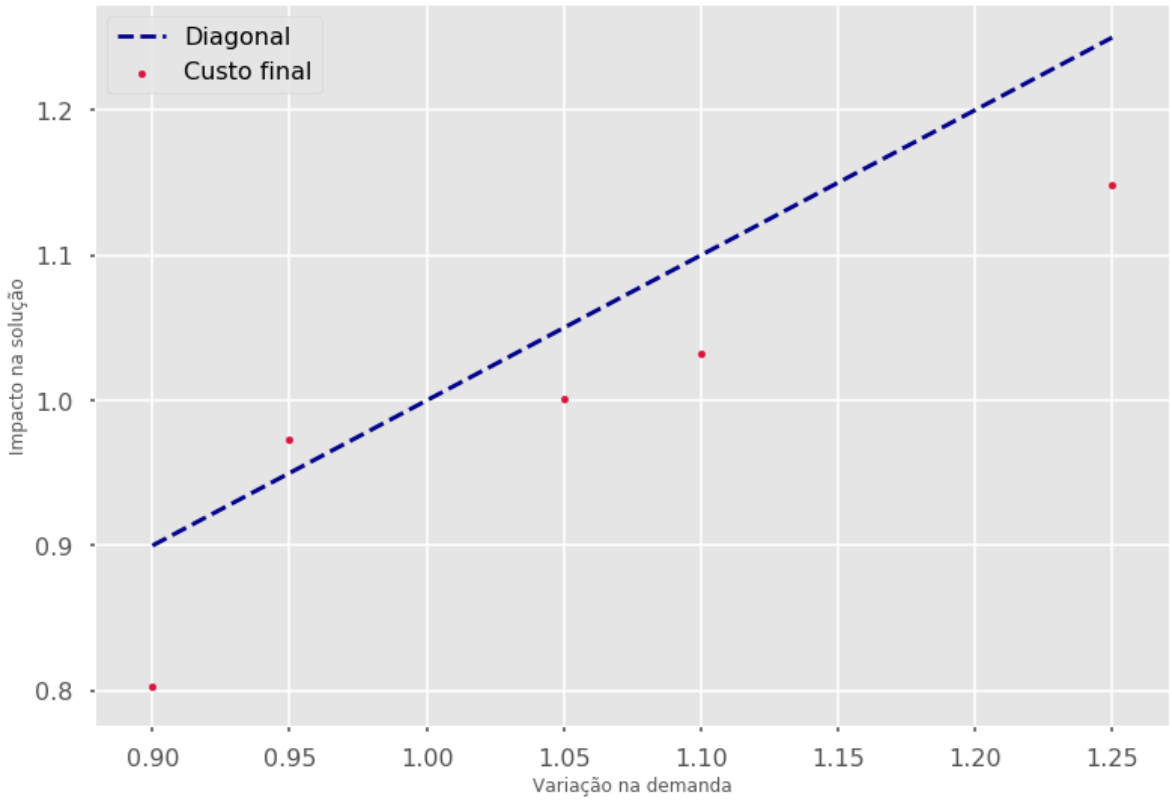
O aumento do custo com o aumento das janelas não foi constatado em testes realizados com o modelo exato em instâncias menores, o que é natural, pois o aumento do espaço de busca de soluções ou melhora ou não altera a solução ótima. O resultado dessa análise de sensibilidade evidencia então o caráter "míope" da heurística construtiva, que não garante a obtenção de uma solução ótima.

Tabela 28: Impacto no aumento das janelas de tempo na solução

Aumento na janela	Variação no custo
1h	+2%
2h	+1.4%

Fonte: elaborado pelo autor

Figura 16: Impacto da variação da demanda no custo final



Fonte: elaborado pelo autor

7.2.4 Variação no tempo de trabalho dos motoristas

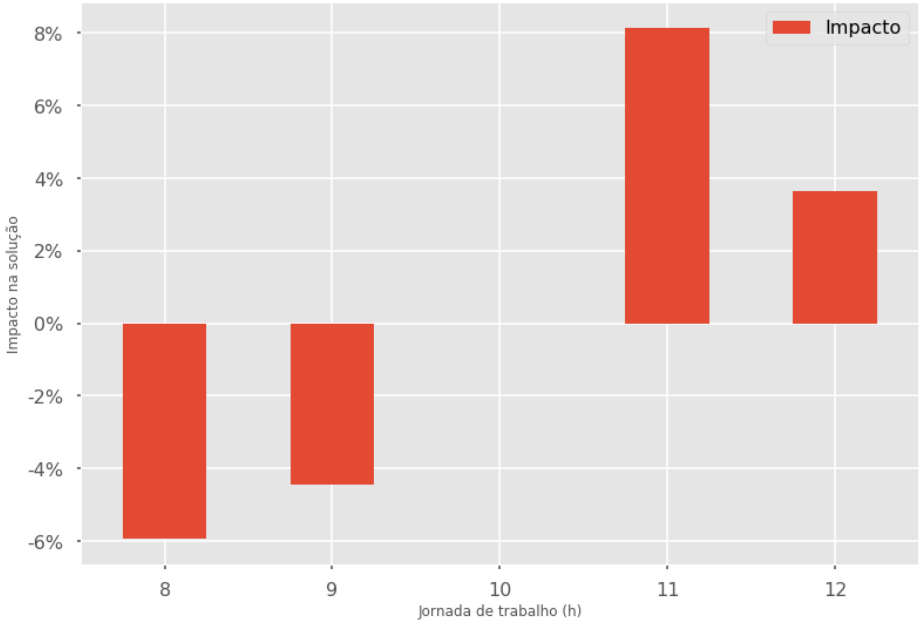
Por fim, o último cenário analisado é a possibilidade de variação no tempo de trabalho dos motoristas. Atualmente a solução considera uma jornada de trabalho de 10 horas e um descanso de 14 horas. Segundo a "Lei do Caminhoneiro" (BRASIL, 2015), a jornada padrão deve durar 8 horas, sendo admitida a realização de 2 a 4 horas extras por dia.

Consideramos então os cenários de trabalho de 8, 9, 10, 11 e 12 horas e como essa variação impacta a formação das rotas. O cenário de referência é o de 10 horas, já que é com base nele que a solução proposta foi construída. Os resultados são apresentados na Figura 17 e os valores na Tabela 29.

É interessante observar que a redução da duração da jornada de trabalho acaba reduzindo o custo total das rotas e o oposto ocorre com o aumento da jornada de trabalho.

Uma possível explicação para esse comportamento é a seguinte: com a redução da jornada de trabalho, a inserção de clientes mais distantes em uma determinada rota se torna menos interessante, pois é mais provável que a inserção implique na necessidade de um período de

Figura 17: Impacto da variação da jornada de trabalho na solução



Fonte: elaborado pelo autor

Tabela 29: Impacto da variação da jornada de trabalho na solução

Jornada de trabalho	Variação no custo
8	-5.9%
9	-4.4%
10	0%
11	8.1%
12	3.6%

Fonte: elaborado pelo autor

descanso para o motorista, o que é penalizado no calculo da função c_{12} dentro da heurística. Isso potencialmente leva à construção de rotas mais curtas (em média).

Como na análise de sensibilidade anterior, este comportamento não foi constatado em testes realizados com o modelo exato em instâncias menores, de acordo com o esperado. O método heurístico apresentado fornece boas soluções para o problema apresentado no trabalho, mas ainda oferece espaço para melhorias, pois não fornece a solução ótima.

A determinação da duração ótima da jornada de trabalho dependerá da instância que está sendo analisada, mas é interessante notar que o modelo e a heurística permitem avaliar esse aspecto do problema.

CONCLUSÃO

Neste trabalho, foi desenvolvida uma metodologia sistemática para a tarefa de roteirização do abastecimento das lojas físicas da Lojas Americanas S.A.

Conforme foi exposto, a operação logística é um elo essencial para a B2W e a LASA enquanto empresas do setor de varejo, e considerando também o contexto de sua expansão de lojas e realização do seu plano estratégico de criação de uma plataforma multicanal de varejo. A criação da LET'S, subsidiária do grupo dedicada à gestão das operações de logística, mostra a importância das operações logísticas para o grupo. A LET'S possui iniciativas de melhorias em diversas etapas do processo de *Fulfillment* e este trabalho contribuiu dentro de uma dessas iniciativas.

Após a realização de uma revisão das técnicas e métodos existentes na literatura para o problema de roteirização de veículos, e a partir do estudo detalhado das características do problema, desenvolveu-se um modelo específico, na forma de um programa de otimização PLIM, para o problema enfrentado pela empresa. Devido ao fato de o problema de roteirização pertencer à classe de complexidade **NP**-difícil, foi necessária a utilização de heurísticas para que fosse possível tentar obter boas soluções de maneira eficiente.

A heurística escolhida e desenvolvida para este trabalho deriva da heurística de inserção inicialmente descrita por Solomon (SOLOMON, 1987) e posteriormente desenvolvida e aprimorada por outros autores. Por ter uma complexidade computacional menor do que o método de solução exata, a heurística permite a resolução em tempo razoável do problema, para instâncias de tamanho variado e de ordem de grandeza condizente com a realidade da empresa.

Não somente a heurística realiza a roteirização de maneira eficiente, como também oferece soluções que aprimoram os roteiros atuais da empresa em termos de custo. Com a aplicação da heurística foi obtido um conjunto de roteiros de custo 12% menor do que a solução atual da empresa. O método apresentado é também robusto sob alguns cenários de *stress* e de possíveis mudanças que a B2W pode sofrer no futuro próximo, fato confirmado por meio de uma série de análises de sensibilidade.

O autor deste trabalho espera então que as soluções e o método propostos sejam utilizados e implementados pela empresa no seu dia a dia, gerando o impacto desejado e aumentando a

produtividade e eficiência dela.

Trabalhos posteriores que podem estender este trabalho incluem: a adição de mais restrições operacionais para o modelo de otimização, considerando por exemplo o problema de roteamento de veículos periódico como problema base; e o estudo de técnicas heurísticas e metaheurísticas alternativas para comparação, visando obter uma solução ainda melhor para o problema.

REFERÊNCIAS

ARENALES, M. *et al. Pesquisa operacional: para cursos de engenharia*. [S.l.]: Elsevier Brasil, 2015.

B2W DIGITAL. *Apresentação de Resultados 1T2019*. [S.l.], 2019. Disponível em: ⟨https://static.b2wdigital.com/upload/apresentacoes/00003189.pdf⟩. Acesso em: 11 Jul. 2019.

B2W DIGITAL. *Relatório de Resultados 1T2019*. [S.l.], 2019. Disponível em: ⟨https://static.b2wdigital.com/upload/releasesderesultados/00003180.pdf⟩. Acesso em: 11 Jul. 2019.

B2W DIGITAL. *Relatório de Resultados 2T2019*. [S.l.], 2019. Disponível em: ⟨https://static.b2wdigital.com/upload/releasesderesultados/00003213.pdf⟩. Acesso em: 28 Out. 2019.

B2W DIGITAL. *Site institucional*. [S.l.], 2019. Disponível em: ⟨https://ri.b2w.digital/institucional/perfil⟩. Acesso em: 11 Jul. 2019.

BALLOU, R. *Business Logistics/Supply Chain Management and Logware*. Prentice Hall PTR, 2003. ISBN 9780131076594. Disponível em: ⟨https://books.google.com.br/books?id=j_rlAQAACAAJ⟩.

BERGSTRA, J.; BENGIO, Y. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, v. 13, n. Feb, p. 281–305, 2012.

BRAEKERS, K.; RAMAEKERS, K.; NIEUWENHUYSE, I. V. The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, Elsevier, v. 99, p. 300–313, 2016.

BRASIL. *LEI Nº 13.103, DE 2 DE MARÇO DE 2015*: Dispõe sobre o exercício da profissão de motorista. Brasília, DF, 2015. Disponível em: ⟨http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2015/lei/l13103.htm⟩. Acesso em: 1 Nov. 2019.

CORMEN, T. H. *et al. Introduction to algorithms*. [S.l.]: MIT press, 2009.

DANTZIG, G.; FULKERSON, R.; JOHNSON, S. Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America*, INFORMS, v. 2, n. 4, p. 393–410, 1954.

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

DROR, M.; LAPORTE, G.; TRUDEAU, P. Vehicle routing with split deliveries. *Discrete Applied Mathematics*, Elsevier, v. 50, n. 3, p. 239–254, 1994.

DULLAERT, W. *et al.* New heuristics for the fleet size and mix vehicle routing problem with time windows. *Journal of the Operational Research Society*, Taylor & Francis, v. 53, n. 11, p. 1232–1238, 2002.

EBIT-NIELSEN. *E-COMMERCE FATURA R$53,2 BILHÕES EM 2018, ALTA DE 12%*. [S.l.], 2019. Disponível em: ⟨https://www.nielsen.com/br/pt/insights/article/2019/e-commerce-fatura-53-bilhoes-em-2018-alta-de-12-por-cento/⟩. Acesso em: 11 Jul. 2019.

EKSIOGLU, B.; VURAL, A. V.; REISMAN, A. The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering*, Elsevier, v. 57, n. 4, p. 1472–1483, 2009.

FUNDAÇÃO DOM CABRAL. *Custos Logísticos no Brasil – 2017*. [S.l.], 2017. Disponível em: ⟨https://www.fdc.org.br/conhecimento-site/nucleos-de-pesquisa-site/Materiais/pesquisa-custos-logisticos2017.pdf⟩. Acesso em: 29 Out. 2019.

ILOS. *Panorama ILOS - Custos Logísticos no Brasil – 2017*. [S.l.], 2017. Disponível em: ⟨https://www.ilos.com.br/web/analise-de-mercado/relatorios-de-pesquisa/custos-logisticos-no-brasil/⟩. Acesso em: 29 Out. 2019.

KRITIKOS, M. N.; IOANNOU, G. The heterogeneous fleet vehicle routing problem with overloads and time windows. *International Journal of Production Economics*, Elsevier, v. 144, n. 1, p. 68–75, 2013.

LAHYANI, R.; KHEMAKHEM, M.; SEMET, F. Rich vehicle routing problems: From a taxonomy to a definition. *European Journal of Operational Research*, Elsevier, v. 241, n. 1, p. 1–14, 2015.

LAPORTE, G. Fifty years of vehicle routing. *Transportation Science*, INFORMS, v. 43, n. 4, p. 408–416, 2009.

LAPORTE, G.; LOUVEAUX, F. V. Solving stochastic routing problems with the integer l-shaped method. In: *Fleet management and logistics*. [S.l.]: Springer, 1998. p. 159–167.

LENSTRA, J. K.; KAN, A. R. Complexity of vehicle routing and scheduling problems. *Networks*, Wiley Online Library, v. 11, n. 2, p. 221–227, 1981.

LOJAS AMERICANAS S.A. *Site institucional*. [S.l.], 2019. Disponível em: ⟨https://ri.lasa.com.br/⟩. Acesso em: 11 Jul. 2019.

MANGUINO, J. L.; RONCONI, D. P. Problema de roteamento de veículos com frota mista, janelas de tempo e custos escalonados. In: *Anais do XLIV Simpósio Brasileiro de Pesquisa Operacional & XVI Congreso Latino-Iberoamericano de Invetigación Operativa*. SOBRAPRO, 2012. Disponível em: ⟨http://www.din.uem.br/sbpo/sbpo2012/⟩. Acesso em: 27 Out. 2019.

PAPADIMITRIOU, C. H.; STEIGLITZ, K. *Combinatorial optimization*. [S.l.]: Prentice Hall Englewood Cliffs, 1982. v. 24.

PARENTE, J. *Varejo no Brasil: gestão e estratégia*. [S.l.]: Editora Atlas, 2000.

PERKEL, J. M. Why jupyter is data scientists' computational notebook of choice. *Nature*, Springer Nature, v. 563, n. 7729, p. 145–146, out. 2018. Disponível em: ⟨https://doi.org/10.1038/d41586-018-07196-1⟩.

PYTHON FOUNDATION. *Python Developers Survey 2018 Results*. [S.l.], 2018. Disponível em: ⟨https://www.jetbrains.com/research/python-developers-survey-2018/⟩. Acesso em: 30 Out. 2019.

RODRIGUEZ, C. M. T. *et al.* A necessidade da participação da logística para a evolução do varejo. *Mundo Logística*, n. 48, p. 32–50, Sep 2015.

RONCONI, D. P.; MANGUINO, J. L. Heurísticas construtivas para o problema de roteamento de veículos com custos escalonados. In: *Anais do XLVIII Simósio Brasileiro de Pesquisa Operacional*. SOBRAPRO, 2016. Disponível em: ⟨http://www.din.uem.br/sbpo/sbpo2016/index.html⟩. Acesso em: 27 Out. 2019.

SINTEF APPLIED MATHEMATICS. *Transportation Optimization Portal - VRPTW - Solomon benchmark*. [S.l.], 2008. Disponível em: ⟨https://www.sintef.no/vrptw⟩. Acesso em: 30 Out. 2019.

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.

SOLOMON, M. M. *VRPTW BENCHMARK PROBLEMS*. [S.l.], 2005. Disponível em: ⟨http://web.cba.neu.edu/~msolomon/problems.htm⟩. Acesso em: 28 Out. 2019.

TOTH, P.; VIGO, D. *Vehicle routing: problems, methods, and applications*. [S.l.]: SIAM, 2014.

VALOR ECONÔMICO. *B2W cria Let's, que unirá logística do grupo e da Lojas Americanas*. [S.l.], 2019. Disponível em: ⟨https://www.valor.com.br/empresas/5370197/b2w-cria-let\%3Fs-que-unira-logistica-do-grupo-e-da-lojas-americanas⟩. Acesso em: 11 Jul. 2019.

VIDAL, T. *et al.* Heuristics for multi-attribute vehicle routing problems: A survey and synthesis. *European Journal of Operational Research*, Elsevier, v. 231, n. 1, p. 1–21, 2013.

WINSTON, W. L.; GOLDBERG, J. B. *Operations research: applications and algorithms*. [S.l.]: Thomson/Brooks/Cole Belmontˆ eCalif Calif, 2004. v. 3.

APÊNDICE A – GERAÇÃO DE INSTÂNCIAS ALEATÓRIAS

Para expandir o universo de testes, definimos a seguir uma metodologia de geração de instâncias aleatórias, criada em conjunto com a empresa. A geração é baseada nos problemas *benchmark* criados por Solomon (SOLOMON, 1987) para o problema de roteirização com janelas de tempo e podem ser encontrados em (SOLOMON, 2005).

Os problemas são separados em grupos R e C: no grupo R as localizações são geradas aleatoriamente, e no grupo C as localizações são *clusterizadas* (agrupadas).

Utilizamos os problemas de classe C, com 100 clientes. No total temos então 2 tipos problemas base (C1 e C2) que podem ser expandidos com a adição das informações particulares do problema tratado nesse trabalho. Caso sejam necessárias instâncias menores, com n clientes ($n < 100$, por exemplo 25 ou 50), basta escolher os n primeiros clientes do problema de 100 clientes correspondente, como descrito em (SINTEF APPLIED MATHEMATICS, 2008). As classes C1 e C2 são bastante similares, com a alteração da localização de um subconjunto de clientes. Nos problemas originais de Solomon, além da localização, também é definida uma frota homogênea e um conjunto de janelas de tempos para os clientes. A classe C1 original possui janelas de tempo mais restritivas e veículos com menor capacidade, enquanto que a classe C2 possui horizontes mais longos e veículos com maior capacidade (SOLOMON, 1987).

Primeiramente, define-se o horizonte de planejamento (número de dias). No que segue, utilizaremos o acrônimo *v.a.u.* para a expressão "variável aleatória uniforme", ou seja, uma variável aleatória de distribuição uniforme. Os parâmetros da distribuição são definidos conforme necessário.

- **Localidades**: Para cada cliente, o início da janela de tempo é uma *v.a.u.* inteira entre 6 e 13 (em horas) e a duração da janela (consequentemente o fim da janela) é uma *v.a.u.* inteira entre 2 e 14.

- **Cargas**: será considerada apenas uma entrega por cliente. O dia de entrega é uma *v.a.u.* inteira dentro do horizonte de planejamento, o volume a ser entregue é dado por $Volume = 0.5 + 49.5 \times U$ onde U é uma *v.a.u.* real entre 0 e 1. O valor da entrega é dado pelo volume correspondente multiplicado por uma *v.a.u.* inteira entre 500 e 2000.

- **Veículos**: consideramos 3 tipos de veículos. O veículo padrão (veículo 1) tem como capacidade uma *v.a.u.* inteira entre 20 e 30, o custo fixo é uma *v.a.u.* inteira entre 50 e 100, o frete é dado por $1 + 2 \times U$ onde U é uma *v.a.u.* real entre 0 e 1. Os outros dois tipos de veículo são definidos a partir do primeiro veículo. A capacidade do veículo 2 é 1.5 vezes a capacidade do veículo 1 e a do veículo 3 é 1.3 vezes a capacidade do veículo 2. O custo fixo do veículo 2 é 1.4 vezes a do veículo 1 e a do veículo 3 é 1.2 vezes a do veículo 2. O frete aumenta em R$ 0.5 com relação ao veículo anterior para cada veículo. A quantidade máxima de veículos para cada tipo é diferente, sendo todas *v.a.u*'s entre 12 e 16. O valor máximo transportado por veículo é o mesmo para os três tipos, dado por 1000 vezes uma *v.a.u.* inteira entre 100 e 200.

- **Malha**: vamos considerar que todas as lojas podem ser abastecidas por todos os veículos. As distâncias dos nós são dadas pelas distâncias cartesianas dos pontos nos problemas base de Solomon, multiplicadas por um fator de escala de 10km:

$$d_{ij} = 10 \times \sqrt{((x_i - x_j)^2 + (y_i - y_j)^2)}$$

  e o tempo trajeto é calculado considerando uma velocidade média constante.

Outros parâmetros são definidos:

- Tempo de início dos veículos: fixados uniformemente de maneira determinística entre 6 e 8 horas da manhã.

- Velocidade média: 80 km/h

Como os dados são de natureza aleatória, é possível que uma instância gerada seja infactível, o que de fato aconteceu durante alguns testes. Para lidar com essa questão de maneira consistente e sistemática, primeiro identificamos as origens da infactibilidade, e em seguida propomos maneiras de tornar factível uma dada instância. As fontes de infatibilidades encontradas foram:

1. Havia lojas com entrega planejada para o primeiro dia; no entanto, a soma do tempo de carregamento no CD com o tempo de trajeto do CD até a loja ultrapassava uma jornada

de trabalho de um motorista, de forma que esta loja não pode ser servida no primeiro dia. De maneira similar, se a soma do tempo de carregamento no CD com o tempo de trajeto do CD até a loja e com o tempo de serviço na loja ultrapassar uma jornada de trabalho, essa loja com entrega planejada no primeiro dia não pode ser atendida.

2. Capacidade insuficiente nos veículos: se o volume a ser entregue para um determinado cliente é maior do que a capacidade do maior veículo.

3. Veículos insuficientes: a soma das capacidades dos veículos não é suficiente para atender toda a demanda dos clientes.

e para torná-las factíveis:

1. Modificar o dia da entrega do cliente para o dia seguinte, mantendo a mesma janela de tempo.

2. Modificar a capacidade do maior veículo para o menor inteiro maior que a maior demanda de uma loja.

3. Aumentar a quantidade de veículos de cada tipo, de forma que tenhamos o mesmo número de veículos por tipo de clientes. Isso superestima a quantidade de veículos total disponível, mas garante que o problema seja factível desse ponto de vista.

APÊNDICE B – CÓDIGO

Neste apêndice mostra-se alguns dos *Jupyter Notebook*'s e códigos utilizados para a implementação e análise dos métodos heurísticos e exato. A base de código pode ser encontrada no seguinte repositório *GitHub*: https://github.com/enochyang/tfpoli-vrptw.

B.1 Método exato

1 VRPTW with mandatory rest, hererogeneous fleet and site dependence

[1]:
```
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

from docplex.mp.model import Model
from docplex.mp.conflict_refiner import ConflictRefiner
import networkx as nx
import time
```

1.1 Data

[2]:
```
data_name = '20191020_1536'
num_instances = 25
solomon_instance = 'c201'
```

[ ]:
```
%run read_data_rand_solomon.py $num_instances $solomon_instance $data_name
```

[6]:
```
G = nx.from_pandas_edgelist(arcs,
                            'start', 'end', ['Distancia (km)','Tempo (h)'],␣
 ↪create_using=nx.DiGraph)
```

1.2 Optimization model

[10]:
```
mdl = Model('VRPTW2')
```

Decision variables

[11]:
```
x_keys = [(i, j, k) for k in fleet_list
                    for (i,j) in arcs[arcs['Veiculo']==fleet_df.
 ↪loc[k]['Veiculos']][['start', 'end']].itertuples(index=False)]
z_keys = [(i, k) for i in start_nodes_list for k in fleet_list]
x = mdl.binary_var_dict(x_keys, name='x')
y = mdl.binary_var_list(fleet_list, name='y')
z = mdl.binary_var_dict(z_keys, name='z')
```

```
Ns = mdl.integer_var_dict(z_keys, name='Ns', lb=0, ub=horizon)
Ws = mdl.continuous_var_dict(z_keys, name='ws', lb=0, ub=day-eps)
Nt = mdl.integer_var_dict(x_keys, name='Nt', lb=0, ub=horizon)
Wt = mdl.continuous_var_dict(x_keys, name='wt', lb=0, ub=work_shift-eps)
s = mdl.continuous_var_matrix(start_nodes_list, fleet_list, name='s', lb=0)
```

Objective function

[12]:
```
start_time = time.perf_counter()
```

[13]:
```
mdl.minimize(mdl.sum(get_dist(i, j, k) * x[(i, j, k)] * freightCost_list[k] for␣
 ↪(i, j, k) in x_keys) +
             mdl.sum(y[k] * fixedCost_list[k] for k in fleet_list))
```

[14]:
```
mdl.add_constraints((mdl.sum(x[0, j, k] for j in customers_list if (0, j, k) in␣
 ↪x_keys)== y[k] for k in fleet_list),
                    names=['vehicle_{}_usage'.format(k) for k in fleet_list])
mdl.add_constraints([mdl.sum(x[i, j, k] for k in fleet_list for j in nodes_list␣
 ↪if j!=0 and j!=i and (i, j, k) in x_keys)==1 for i in customers_list],
                    names=['visit_customer_{}'.format(i) for i in␣
 ↪customers_list])
mdl.add_constraints([mdl.sum(demand_volume[i] *
                             mdl.sum(x[i, j, k] for j in end_nodes_list if j!=i␣
 ↪and (i, j, k) in x_keys)
                             for i in customers_list) <= QVolume_list[k] * y[k]␣
 ↪for k in fleet_list],
                    names=['volume_capacity_{}'.format(k) for k in fleet_list])
mdl.add_constraints([mdl.sum(demand_value[i] *
                            mdl.sum(x[i, j, k] for j in end_nodes_list if j!=i␣
 ↪and (i, j, k) in x_keys)
                             for i in customers_list) <= QValue_list[k] * y[k]␣
 ↪for k in fleet_list],
                    names=['value_capacity_{}'.format(k) for k in fleet_list])
mdl.add_constraints([mdl.sum(x[0, j, k] for j in end_nodes_list  if (0, j, k)␣
 ↪in x_keys)==1 for k in fleet_list],
                     names=['start_from_CD_{}'.format(k) for k in fleet_list])
mdl.add_constraints([mdl.sum(x[i, h, k] for i in start_nodes_list if i!=h and␣
 ↪(i, h, k) in x_keys) ==
                     mdl.sum(x[h, j, k] for j in end_nodes_list if j!=h and (h,␣
 ↪j, k) in x_keys) for h in customers_list
                     for k in fleet_list],
                    names=['flow_constraint_{}_{}'.format(h, k) for h in␣
 ↪customers_list for k in fleet_list])
mdl.add_constraints([mdl.sum(x[i, n+1, k] for i in start_nodes_list if (i, n+1,␣
 ↪k) in x_keys)==1 for k in fleet_list],
                    names=['end_in_CD_{}'.format(k) for k in fleet_list])
mdl.add_constraints([start_windows[i] <= s[i, k] for i in customers_list for k␣
 ↪in fleet_list],
```

```
                    names=['left_window_{}_{}'.format(i, k) for i in␣
 ↪customers_list for k in fleet_list])
mdl.add_constraints([s[i, k] <= end_windows[i] for i in customers_list for k in␣
 ↪fleet_list],
                    names=['right_window_{}_{}'.format(i, k) for i in␣
 ↪customers_list for k in fleet_list])
mdl.add_constraints([s[0, k] == vehicle_start_hour[k] for k in fleet_list],
                    names=['start_time_from_CD_{}'.format(k) for k in␣
 ↪fleet_list])
mdl.add_constraints([s[i, k] + get_time(i, j, k) + service_time[i] <= s[j, k] +␣
 ↪(1 - x[i, j, k])*get_M(i, j, k)
                     for i in start_nodes_list for j in customers_list for k in␣
 ↪fleet_list if i!=j and (i, j, k) in x_keys],
                    names=['service_continuity_{}_{}_{}'.format(i, j, k) for i␣
 ↪in start_nodes_list for j in customers_list
                           for k in fleet_list if i!=j and (i, j, k) in x_keys])
mdl.add_constraints([(s[i, k] + service_time[i] - vehicle_start_hour[k]) ==␣
 ↪((Ns[i, k]*day) + Ws[i, k]) for i in start_nodes_list
                     for j in end_nodes_list for k in fleet_list if i!=j and␣
 ↪(i, j, k) in x_keys],
                     names=['working_hours_{}_{}'.format(i, k) for i in␣
 ↪start_nodes_list for j in end_nodes_list
                            for k in fleet_list if i!=j and (i, j, k) in␣
 ↪x_keys])
mdl.add_indicator_constraints([mdl.indicator_constraint(x[i, j, k], Ws[i, k] +␣
 ↪get_time(i, j, k) - work_shift + eps <= Mp * (1 - z[i, k]),
                                                         active_value=1,␣
 ↪name='need_rest_after_{}_{}_{}'.format(i, j, k))
                               for i in start_nodes_list for j in customers_list␣
 ↪for k in fleet_list if i!=j and (i, j, k) in x_keys])
mdl.add_indicator_constraints([mdl.indicator_constraint(x[i, j, k], Ws[i, k] +␣
 ↪get_time(i, j, k) == Nt[i, j, k] * work_shift + Wt[i, j, k],
                                                         active_value=1,␣
 ↪name='rest_time_{}_{}_{}'.format(i, j, k))
                               for i in start_nodes_list for j in customers_list␣
 ↪for k in fleet_list if i!=j and (i, j, k) in x_keys])
mdl.add_indicator_constraints([mdl.indicator_constraint(x[i, j, k], ((s[i, k] +␣
 ↪service_time[i] + get_time(i, j, k) + Nt[i, j, k] * rest_shift) - s[j, k])␣
 ↪<= Mp * z[i, k],
                                                         active_value=1,␣
 ↪name='serve_after_rest_{}_{}_{}'.format(i, j, k))
                               for i in start_nodes_list for j in customers_list␣
 ↪for k in fleet_list if i!=j and (i, j, k) in x_keys])
print("Added constraints")
```

[15]:
```
end_time = time.perf_counter()
obj_cst_time = end_time - start_time
print(obj_cst_time)
```

```
791.5650526000001
```

[16]:
```
# both limited to 1000 in trial version
print("Number of constraints: " + str(mdl.number_of_constraints))
print("Number of variables: " + str(mdl.number_of_variables))
```

```
Number of constraints: 174553
Number of variables: 111132
```

[ ]:
```
solution = mdl.solve(log_output=True)
print(mdl.solve_details)
print(solution.solve_status)
```

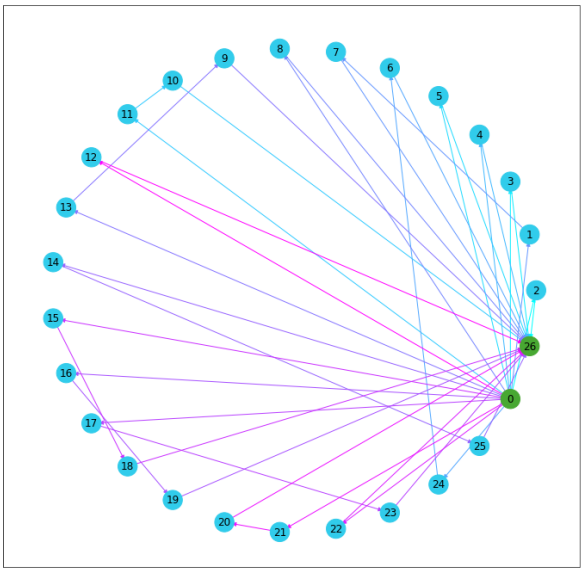
[24]:
```
active_vehicles = [i for i in fleet_list if y[i].solution_value>0]
active_arcs = [(i, j, k) for (i, j, k) in x_keys if x[i, j, k].solution_value>0]
routes = {v:[(i, j) for i, j, k in active_arcs if k==v] for v in␣
 ↪active_vehicles}
svc_times = {k:{i: s[i, k].solution_value for i, j in routes[k] if i in␣
 ↪nodes_list} for k in active_vehicles}
```

[25]:
```
print("Objective value: " + str(solution.objective_value))
print("Service times: " + str(svc_times))
print("Active vehicles: " + str(active_vehicles))
# vehicles
print("Vehicle types: " + str([fleet_df['Veiculos'].iloc[k] for k in␣
 ↪active_vehicles]))
print("Routes: " + str(routes))
```

```
Objective value: 26297.7051776577
Service times: {1: {0: 7.0, 2: 59.99999999999999}, 16: {0: 7.0, 3: 73.0}, 17:
{0: 8.0, 5: 59.999999999999986}, 18: {11: 35.0, 0: 6.0, 10: 54.0}, 20: {0: 8.0,
4: 37.0}, 22: {24: 10.475000000000023, 0: 7.0, 6: 13.0}, 24: {0: 6.0, 1:
21.999999999999996, 7: 37.0}, 27: {0: 6.0, 8: 72.0}, 30: {13: 35.0, 0: 6.0, 9:
57.0}, 31: {0: 7.0, 14: 56.0, 25: 60.82499999999999}, 39: {0: 6.0, 16: 35.0, 19:
36.46721680981652}, 42: {0: 6.0, 17: 12.01789277477549, 23: 63.00000000000001},
45: {0: 6.0, 15: 36.00000000000001, 18: 37.88388347648319}, 46: {0: 7.0, 22:
18.0}, 47: {21: 31.0, 0: 8.0, 20: 47.0}, 51: {0: 6.0, 12: 48.0}}
Active vehicles: [1, 16, 17, 18, 20, 22, 24, 27, 30, 31, 39, 42, 45, 46, 47, 51]
Vehicle types: ['veiculo1', 'veiculo2', 'veiculo2', 'veiculo2', 'veiculo2',
'veiculo2', 'veiculo2', 'veiculo2', 'veiculo3', 'veiculo3', 'veiculo3',
'veiculo3', 'veiculo3', 'veiculo3', 'veiculo3', 'veiculo3']
Routes: {1: [(0, 2), (2, 26)], 16: [(0, 3), (3, 26)], 17: [(0, 5), (5, 26)], 18:
[(11, 10), (0, 11), (10, 26)], 20: [(0, 4), (4, 26)], 22: [(24, 6), (0, 24), (6,
```

```
26)], 24: [(0, 1), (1, 7), (7, 26)], 27: [(0, 8), (8, 26)], 30: [(13, 9), (0,
13), (9, 26)], 31: [(0, 14), (14, 25), (25, 26)], 39: [(0, 16), (16, 19), (19,
26)], 42: [(0, 17), (17, 23), (23, 26)], 45: [(0, 15), (15, 18), (18, 26)], 46:
[(0, 22), (22, 26)], 47: [(21, 20), (0, 21), (20, 26)], 51: [(0, 12), (12, 26)]}
```

[26]:
```
plt.figure(figsize=(12, 12))
values = ['#31cceb' if node!=0 and node!=n+1 else '#48a832' for node in G.nodes]
#pos = nx.kamada_kawai_layout(G, pos=nx.get_node_attributes(G,'pos'),␣
 ↪center=(0, 0)) #weight='distance'
#pos = nx.spring_layout(G, k=50, pos=nx.get_node_attributes(G,'pos'),␣
 ↪weight='distance')
pos = nx.circular_layout(G, scale=1, center=(0,0))
nx.draw_networkx_nodes(G, pos, cmap=plt.get_cmap('jet'),
                       node_color = values, node_size = 500)
nx.draw_networkx_labels(G, pos)
color_idx = np.linspace(0, 1, len(routes.items()))
for i, (vehicle, edges) in enumerate(routes.items()):
    nx.draw_networkx_edges(G, pos, edgelist=edges, edge_color=matplotlib.colors.
 ↪to_hex(plt.cm.cool(color_idx[i])), arrows=True)
    edge_distances = nx.get_edge_attributes(G, 'Distancia (km)')
    edge_distances_to_draw = {e:edge_distances[e] for e in edges}
    #nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_distances_to_draw)
#plt.show()
plt.savefig(os.path.join(dir_path, "comparisons",␣
 ↪"routes","figures",'cplex_routes_'+data_name+'_'+str(num_instances)+'_'+solomon_instance+'.
 ↪png'))
```



B.2 Método heurístico

1 Insertion Heuristic for the VRPTW with heterogeneous fleet, site dependence and rest times

[1]:
```
from datetime import datetime
import functools
import time
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import networkx as nx
```

[2]:
```
from operator import itemgetter
```

[3]:
```
import itertools
def pairwise(iterable):
    "s -> (s0,s1), (s1,s2), (s2, s3), ..."
    a, b = itertools.tee(iterable)
    next(b, None)
    return zip(a, b)
```

1.1 Data

[4]:
```
%run read_data_real_v4.py
```

[5]:
```
G = nx.from_pandas_edgelist(arcs,
                            'start', 'end', ['Distancia (km)','Tempo (h)'],␣
 ↪create_using=nx.DiGraph)
Gs = [nx.from_pandas_edgelist(arcs[arcs['Veiculo']==vehicle],
                            'start', 'end', ['Distancia (km)','Tempo (h)'],␣
 ↪create_using=nx.DiGraph)
     for vehicle in capacidades['Veiculos'].unique()]
```

1.2 Heuristic

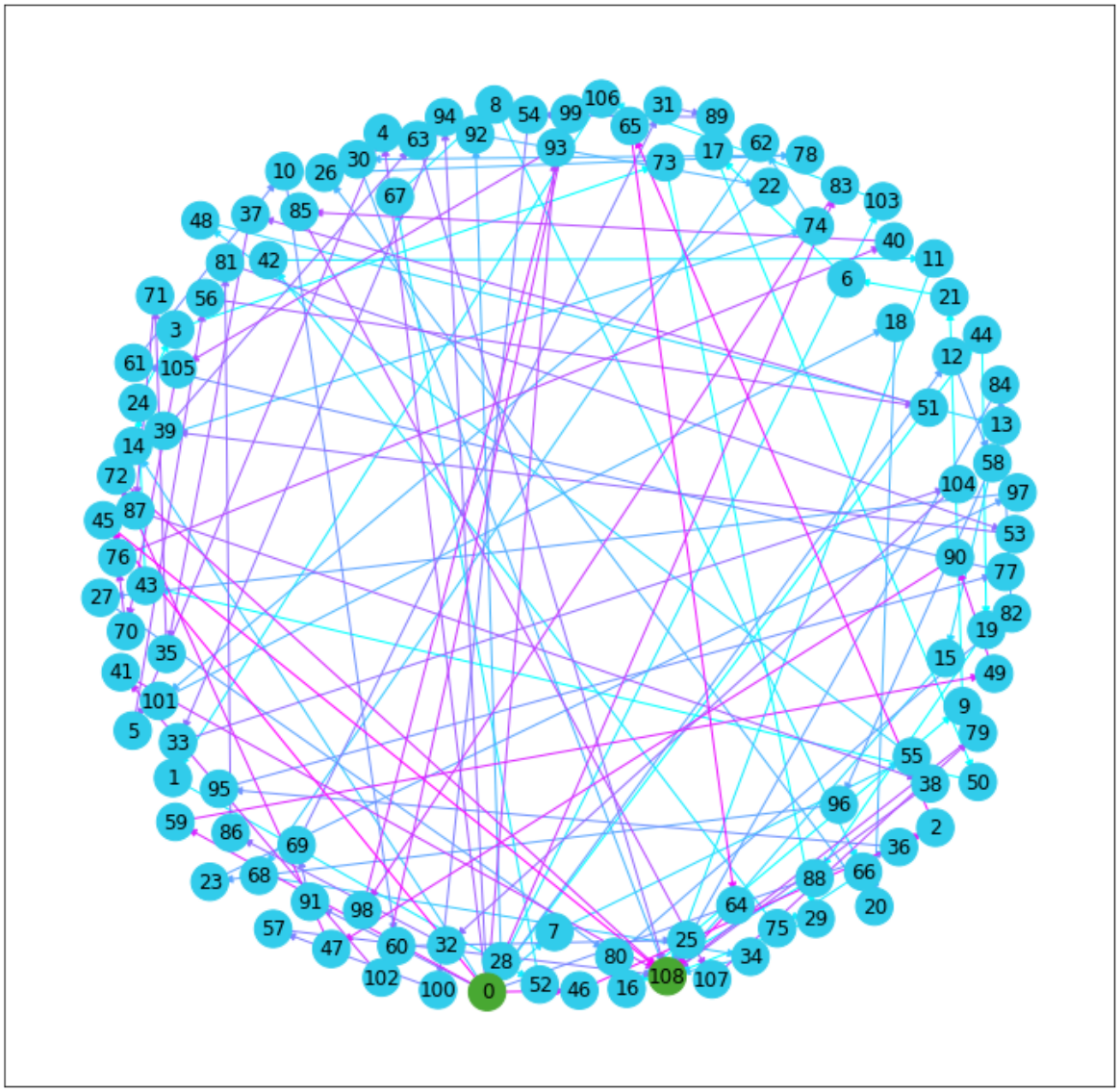
[6]:
```
import math
```

[7]:
```
%run -i insertion_heuristic_real_data.py
```

[8]:
```
routes_non_norm_gs, run_time = insertion_heuristic(weights_c1 = [0.496324, 0.
 ↪6985, 0.5], weights_c2 = [0.5294, 0.5956, 0.5368])
print("Total cost: "+str(sum([r.compute_route_cost() for r in␣
 ↪routes_non_norm_gs])))
routes_non_norm_gs_df = routes_to_df(routes_non_norm_gs)
print(compute_cost_routes_df(routes_non_norm_gs_df))
routes_non_norm_gs_df.to_csv('solutions/sol_v4_non_norm_gs.csv')
```

```
Finished 'insertion_heuristic' in 1384.5868 secs
Total cost: 54956.86947176141
54956.86947176141
```

[11]:
```
plt.figure(figsize=(12, 12))
values = ['#31cceb' if node!=0 and node!=n+1 else '#48a832' for node in G.nodes]
#pos = nx.kamada_kawai_layout(G, pos=nx.get_node_attributes(G,'pos'),␣
 ↪center=(0, 0)) #weight='distance'
pos = nx.spring_layout(G, k=50, weight='distance')
#pos = nx.circular_layout(G, scale=1, center=(0,0))
nx.draw_networkx_nodes(G, pos, cmap=plt.get_cmap('jet'),
                       node_color = values, node_size = 500)
nx.draw_networkx_labels(G, pos)
color_idx = np.linspace(0, 1, len(routes_non_norm_gs))
for i, r in enumerate(routes_non_norm_gs):
    edges = [(a, b) for (a, b) in pairwise(r.clients)]
    nx.draw_networkx_edges(G, pos, edgelist=edges, edge_color=matplotlib.colors.
 ↪to_hex(plt.cm.cool(color_idx[i])), arrows=True)
    edge_distances = nx.get_edge_attributes(G, 'Distancia (km)')
    edge_distances_to_draw = {e:edge_distances[e] for e in edges}
    #nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_distances_to_draw)
plt.show()
```



B.3 Código da heurística

```
'''
Heuristica de insercao para o problema de roteirizacao de veiculos

Caracteristicas: janelas de tempo, frota heterogenea, descanso obrigatorio
TODO: refactor, separar as fun  es em diferentes arquivos

author: enochyang
'''

from operator import itemgetter
import math

```

```
13 import functools
14 import time
15 import pandas as pd
16
17
18 def timer(func):
19     '''Print the execution time of the decorated function'''
20     @functools.wraps(func)
21     def wrapper_timer(*args, **kwargs):
22         start_time = time.perf_counter()
23         value = func(*args, **kwargs)
24         end_time = time.perf_counter()
25         run_time = end_time - start_time
26         print(f"Finished {func.__name__!r} in {run_time:.4f}
               secs")
27         return value, run_time
28     return wrapper_timer
29
30
31 def get_vehicle(clients, verbose=False):
32     '''get the vehicle with lowest fixed cost that can serve the
           clients'''
33     total_client_volume = sum([demand_volume[c] for c in clients
           ])
34     total_client_value = sum([demand_value[c] for c in clients])
35     early_start_time = compute_early_start_time(clients)
36     late_start_time = compute_late_start_time(clients)
37     for k in sorted(fleet_list, key=lambda x: fixedCost_list[x])
           :
38         if (not vehicle_in_use[k] and #QWeight[k] >=
               demand_weight and
39                 QVolume_list[k] >= total_client_volume and
40                 QValue_list[k] >= total_client_value):
41             accepted = [vehicle_acceptance[c][k] for c in
                   clients]
42             if not all(accepted):
43                 continue
44             try:
45                 get_service_times_(clients, k, verbose)
46                 return k
47             except:
48                 continue
49     if verbose:
50         print("ERROR: No vehicle available to serve these
               clients: "+str(clients))
51         print("Total volume: "+str(total_client_volume))
52         print("Total value: "+str(total_client_value))
53     assert False, "No vehicles"
54     return
55
56
```

```
57 def get_largest_vehicle_available(clients=[], verbose=False):
58     '''get the largest vehicle that can serve the clients'''
59     candidates = []
60     for k in fleet_list:
61         if not vehicle_in_use[k]:
62             accepted = [vehicle_acceptance[c][k] for c in
                   clients]
63             if all(accepted):
64                 candidates.append(k)
65     if not candidates:
66         if verbose:
67             print("largest vehicle function failed, no vehicles
                   are available")
68         return None
69     volume_capacities = [QVolume_list[k] for k in candidates]
70     value_capacities = [QValue_list[k] for k in candidates]
71     # assuming that the limiting factor is the volume
72     return candidates[volume_capacities.index(max(
           volume_capacities))]
73
74
75 def get_service_times_(clients, vehicle, verbose=False):
76     sh = vehicle_start_hour[vehicle]
77     svc_times = [sh for _ in clients] # all start at time of
           vehicle
78     for i, c in enumerate(clients):
79         if c == 0:
80             continue
81         sjk = (svc_times[i-1] + service_time[clients[i-1]]
82                + get_time(clients[i-1], c, vehicle))
83         if ((svc_times[i-1] + service_time[clients[i-1]] - sh)%
               day
84             + get_time(clients[i-1], c, vehicle) > work_shift):
85             sjk += (rest_shift *
86                     int(((svc_times[i-1] + service_time[clients[
                           i-1]] - sh)%day
87                          + get_time(clients[i-1], c, vehicle)) /
                             work_shift))
88         if sjk > end_windows[c]:
89             if verbose:
90                 print("This route is infeasible, time window
                       violation on client "
91                       +str(c)+" in position "+str(i))
92             assert False
93         svc_times[i] = max(start_windows[c], sjk)
94         if (svc_times[i] + service_time[c] - sh)%day >
               work_shift:
95             if verbose:
96                 print("This route is infeasible, work shift
                       violation on client "
97                       +str(c)+" in position "+str(i))
```

```
98                assert False
99        return svc_times
100
101
102   class Route:
103       '''Route object that holds clients, a vehicle and can return
              some parameters of the route'''
104       def __init__(self, client, fake=False):
105           self.clients = [0, client, n+1]
106           self.vehicle = get_vehicle(self.clients)
107           if not fake:
108               vehicle_in_use[self.vehicle] = True
109           self.update_route_capacity()
110           self.service_times = self.get_service_times(self.clients
                 )
111
112       def __repr__(self):
113           route_str = "Clients in route: "+str(self.clients)+"\n"
114           route_str += "Vehicle: "+str(self.vehicle)+"\n"
115           route_str += "Vehicle type: "+str(fleet_df['Veiculos'].
                 iloc[self.vehicle])+"\n"
116           route_str += "Volume used: "+str(self.total_volume)+"/"+
                 str(self.volume_capacity)+"\n"
117           route_str += "Value used: "+str(self.total_value)+"/"+
                 str(self.value_capacity)+"\n"
118           route_str += "Total distance: "+str(self.
                 get_total_distance())+"\n"
119           route_str += "Service times: "+str(self.
                 get_service_times(self.clients))
120           return route_str
121
122       def update_route_capacity(self):
123           self.volume_capacity = QVolume_list[self.vehicle]
124           self.value_capacity = QValue_list[self.vehicle]
125           self.total_volume = 0
126           self.total_value = 0
127           for c in self.clients:
128               self.total_volume += demand_volume[c]
129               self.total_value += demand_value[c]
130           assert (self.volume_capacity >= self.total_volume), "
                 Volume capacity violation"
131           assert (self.value_capacity >= self.total_value), "Value
                  capacity violation"
132
133       def check_capacity(self, client):
134           # check if client fits in route
135           return ((self.volume_capacity >= demand_volume[client])
                 and
136                   (self.value_capacity >= demand_value[client]))
137
138       def insert_client(self, client, i, verbose=False):
```

```
139         '''Insert client in position i'''
140         assert self.check_capacity(client), "Capacity violation"
141         assert i > 0, "Cannot insert in position 0"
142         assert i < n+1, "Cannot insert in last position"
143         assert get_dist(self.clients[i-1], client, self.vehicle)
                is not None, "No route between previous client and
               this one for the current vehicle"
144         assert get_dist(client, self.clients[i], self.vehicle)
               is not None, "No route between previous client and
               this one for the current vehicle"
145         self.get_service_times(self.clients[0:i]+[client]+self.
               clients[i:]) # assert feasible
146         self.clients.insert(i, client)
147         try:
148             self.update_route_capacity()
149         except:
150             vehicle_in_use[self.vehicle] = False
151             self.vehicle = get_vehicle(self.clients)
152             vehicle_in_use[self.vehicle] = True
153             self.update_route_capacity()
154         self.service_times = self.get_service_times(self.clients
               )
155
156     def get_total_distance(self):
157         total_dist = 0
158         for i in range(len(self.clients)-1):
159             total_dist += get_dist(self.clients[i], self.clients
                   [i+1], self.vehicle)
160         return total_dist
161
162     def get_service_times_old(self, clients, verbose=False):
163         svc_times = [0 for _ in clients] # all start at 0
164         for i, c in enumerate(clients):
165             if c == 0:
166                 continue
167             else:
168                 sjk = svc_times[i-1] + service_time[i-1] +
                       get_time(clients[i-1], c, self.vehicle)
169                 if (svc_times[i-1] + service_time[i-1])%day +
                       get_time(clients[i-1], c, self.vehicle) >
                       work_shift:
170                     sjk += rest_shift*int(((svc_times[i-1] +
                           service_time[i-1])%day + get_time(clients
                           [i-1], c, self.vehicle)) / work_shift)
171                 if sjk > end_windows[c]:
172                     if verbose:
173                         print("This route is infeasible, time
                               window violation on client "+str(c))
174                     assert False
175                 svc_times[i] = max(start_windows[c], sjk)
176         return svc_times
```

```
177
178     def get_service_times(self, clients, verbose=False):
179         return get_service_times_(clients, self.vehicle, verbose
                )
180
181     def compute_route_cost(self):
182         dists = 0
183         for i, c in enumerate(self.clients):
184             if c == 0:
185                 continue
186             else:
187                 dists += get_dist(self.clients[i-1], self.
                        clients[i], self.vehicle)
188         return freightCost_list[self.vehicle]*dists +
                fixedCost_list[self.vehicle]
189
190     def to_dataframe(self):
191         self.service_times = self.get_service_times(self.clients
                )
192         records = []
193         for i, client in enumerate(self.clients):
194             singleton = {'Veículo': fleet_df['Veiculos'].iloc[
                    self.vehicle],
195                          'ID Veiculo': self.vehicle,
196                          'Sequencia': i,
197                          'index': client,
198                          'Loja': nodes.loc[nodes['index'] ==
                             client, 'Localidade'].iloc[0],
199                          'Horario de servi o': self.
                             service_times[i],
200                          'Dia Chegada': int(self.service_times[i
                             ] // day),
201                         }
202             records.append(singleton)
203         return pd.DataFrame.from_records(records)
204
205
206 ##################
207 ##### Costs  ####
208 ##################
209
210 def compute_c11(route, client, i, verbose=False):
211     '''Increase in distance from adding client in position i+1
            '''
212     return (get_dist(route.clients[i], client, route.vehicle) +
213             get_dist(client, route.clients[i+1], route.vehicle)
                -
214             get_dist(route.clients[i], route.clients[i+1], route
                .vehicle))
215
216 def compute_c12(route, client, i, verbose=False):
```

```
217     # tries to compute the service time increase. If infeasible
            will assert error, and then we return infinity
218     try:
219         hyp_svc_times = route.get_service_times(route.clients[0:
                i+1]+[client]+route.clients[i+1:], verbose)[i+1+1] #
                we insert new client in i+1, so previous i+1 becomes
                i+2
220     except:
221         if verbose:
222             print("c12 infinity")
223         hyp_svc_times = math.inf
224     return hyp_svc_times - route.service_times[i+1]
225
226 def compute_c13(route, client, i, verbose=False):
227     '''If capacity is not enough, return increase in fixed cost
            due to change in vehicle'''
228     if route.check_capacity(client):
229         return 0
230     else:
231         try:
232             new_vehicle = route.get_vehicle(route.clients+[
                    client], verbose)
233         except:
234             if verbose:
235                 print("c13 infinity")
236             return math.inf
237         return (fixedCost_list[new_vehicle] - fixedCost_list[
                route.vehicle] +
238                 (freightCost_list[new_vehicle] -
                    freightCost_list[route.vehicle]) * route.
                    get_total_distance())
239
240 def compute_c1(route, client, weights, verbose=False):
241     min_cost = math.inf
242     position = 0
243     for i in range(len(route.clients)-1):
244         new_cost = (weights[0]*compute_c11(route, client, i,
                verbose) +
245                     weights[1]*compute_c12(route, client, i,
                        verbose) +
246                     weights[2]*compute_c13(route, client, i,
                        verbose))
247         if new_cost < min_cost:
248             min_cost = new_cost
249             position = i+1
250     return (min_cost, position)
251
252 def compute_c2(route, client, weights_c1, weights_c2, verbose=
    False):
253     big_distance = math.inf
254     c1, position = compute_c1(route, client, weights_c1, verbose
```

```
        )
    try:
        new_route = Route(client, fake=True)
        c2 = (weights_c2[0]*new_route.get_total_distance() +
              weights_c2[1]*(get_time(0, client, new_route.
                  vehicle) +
                             new_route.service_times[1]) +
              weights_c2[2]*(fixedCost_list[new_route.vehicle] +
                  freightCost_list[new_route.vehicle]*new_route.
                  get_total_distance())
              - c1)
    except:
        if verbose:
            print("Error computing c2 for this client, probably
                we cannot create a new route")
        if math.isinf(c1):
            if verbose:
                print("c1 is infinite in this case")
            c2 = -math.inf
        else:
            if verbose:
                print("c1 is finite in this case")
            c2 = math.inf
    return client, c2, c1, position


@timer
def insertion_heuristic(weights_c1, weights_c2, verbose=False):
    global vehicle_in_use
    vehicle_in_use = [False for k in fleet_list]
    routes = []
    customer_routed = [True] + [False for c in customers_list] +
        [True]
    while not all(customer_routed):
        unrouted = [i for i, c in enumerate(customer_routed) if
            not customer_routed[i]]
        unrouted_dist = [max([get_dist(0, i, k) for k in
            fleet_list])
                         for i in unrouted]
        if verbose:
            print("Unrouted clients total: ", len(unrouted))
        target_client = unrouted[unrouted_dist.index(max(
            unrouted_dist))]
        route = Route(target_client)
        unrouted.remove(target_client)
        customer_routed[target_client] = True
        if verbose:
            print("Target client: ", target_client)
            print("Vehicles in use: ", vehicle_in_use)
        largest_vehicle = get_largest_vehicle_available(clients=
            route.clients, verbose=verbose)
```

```
296         if not largest_vehicle:
297             largest_vehicle = route.vehicle
298         QVolume = QVolume_list[largest_vehicle] - route.
               total_volume
299         QValue = QValue_list[largest_vehicle] - route.
               total_value
300         candidates = [c for c in unrouted if
301                       (#(QWeight >= sum([d * m for d, m in zip(
                              demand[c], weight)])) and
302                        (QVolume >= demand_volume[c]) and
303                        (QValue >= demand_value[c]) and
304                        vehicle_acceptance[c][route.vehicle])]
305         if verbose:
306             print("Candidates:", candidates)
307         while len(candidates) > 0:
308             C2 = [compute_c2(route, c, weights_c1, weights_c2,
                       verbose) for c in candidates]
309             pC2 = [(client, c2, c1, position) for client, c2, c1
                       , position in C2 if c2 > 0]
310             if len(pC2) > 0:
311                 client, _, _, position = max(pC2, key=itemgetter
                           (1))
312                 if verbose:
313                     print("C2:", pC2)
314                     print("Selected client: ", client, " at
                               position ",position)
315                 route.insert_client(client, position, verbose)
316                 unrouted.remove(client)
317                 if verbose:
318                     print("Routed client: ", client)
319                     print("Current route:", route)
320                     print("Vehicles in use: ", vehicle_in_use)
321                 customer_routed[client] = True
322                 largest_vehicle = get_largest_vehicle_available(
                           clients=route.clients, verbose=verbose)
323                 if not largest_vehicle:
324                     largest_vehicle = route.vehicle
325                 QVolume = QVolume_list[largest_vehicle] - route.
                           total_volume
326                 QValue = QValue_list[largest_vehicle] - route.
                           total_value
327                 candidates = [c for c in unrouted if
328                               ((QVolume >= demand_volume[c]) and
329                                (QValue >= demand_value[c]) and
330                                vehicle_acceptance[c][route.
                                       vehicle])]
331                 if verbose:
332                     print("Candidades: ", candidates)
333             else:
334                 if verbose:
335                     print("No candidates for current route,
```

```
                        starting new one")
336                 break
337         routes.append(route)
338         if verbose:
339             print("Latest ROUTE: ", routes[-1])
340     return routes
341
342
343 time_const = arcs['Tempo (h)'].max()
344 dist_const = arcs['Distancia (km)'].max()
345 cf_const = capacidades['Custo Fixo'].max()
346 cv_const = capacidades['Frete Km'].max()
347
348 def normalize_tables(time_const=time_const, dist_const=
       dist_const, cf_const=cf_const, cv_const=cv_const):
349     global service_time
350     global arcs
351     global work_shift
352     global rest_shift
353     global day
354     global vehicle_start_hour
355     global start_windows
356     global end_windows
357     global fixedCost_list
358     global freightCost_list
359     service_time /= time_const
360     arcs['Tempo (h)'] /= time_const
361     work_shift /= time_const
362     rest_shift /= time_const
363     day /= time_const
364     start_windows /= time_const
365     end_windows /= time_const
366     vehicle_start_hour /= time_const
367     arcs['Distancia (km)'] /= dist_const
368     fixedCost_list /= cf_const
369     freightCost_list /= cv_const
370
371
372 def denormalize_tables():
373     normalize_tables(1/time_const, 1/dist_const, 1/cf_const, 1/
          cv_const)
374
375
376 def routes_to_df(routes):
377     df_list = []
378     for i, r in enumerate(routes):
379         df_r = r.to_dataframe()
380         df_r['ID Viagem'] = i + 1
381         df_list.append(df_r)
382     return pd.concat(df_list, axis=0)
383
```

```
384
385 def compute_cost_route_idx(routes_df, idx, add_cd=False):
386     route = routes_df[routes_df['ID Viagem'] == idx]
387     vehicle = route['Veículo'].unique()[0]
388     cf = capacidades.loc[capacidades['Veiculos'] == vehicle, '
        Custo Fixo'].iloc[0]
389     clients = list(route.sort_values('Sequencia')['index'].
        values)
390     if add_cd:
391         clients = [0] + clients + [n+1]
392     dist = sum([get_dist(i, j, vehicle)
393                 for i, j in pairwise(clients)])
394     cv = dist*capacidades.loc[capacidades['Veiculos'] == vehicle
        , 'Frete Km'].iloc[0]
395     return cf + cv
396
397
398 def compute_cost_routes_df(routes_df, add_cd=False):
399     return sum([compute_cost_route_idx(routes_df, idx, add_cd)
        for idx in routes_df['ID Viagem'].unique()])
```

APÊNDICE C – NOVO ROTEIRO PROPOSTO

Aqui apresentamos as rotas do roteiro proposto à B2W, que podem ser vistas na Tabela 30.

Tabela 30: Roteiro otimizado proposto

ID Viagem	Rota
1	CD, 103, 077, 058, 074, 062, 006, 017, 043, CD
2	CD, 044, 008, 019, 088, 016, 067, 052, 001, 020, 026, 078, 010, 060, 025, 072, CD
3	CD, 028, 021, 055, 013, 048, CD
4	CD, 031, 014, 069, 101, 022, 089, 100, 085, 107, CD
5	CD, 061, 068, 034, 015, CD
6	CD, 056, 005, 051, 037, 032, 087, 038, 023, 099, CD
7	CD, 004, 096, 035, 082, 080, 027, 097, 086, 102, 084, CD
8	CD, 049, 104, 033, 094, 106, 039, 053, 081, 063, 030, CD
9	CD, 083, 040, 079, 090, 047, CD
10	CD, 095, 036, 007, 041, CD
11	CD, 093, 091, 092, 105, 054, 057, CD
12	CD, 002, 059, 098, CD
13	CD, 018, 050, 024, 003, 073, 029, 064, CD
14	CD, 046, 070, 009, 066, 075, 042, 011, CD
15	CD, 045, 012, CD
16	CD, 065, 071, CD
17	CD, 076, CD

Fonte: elaborado pelo autor