Workshop PBP-Loggi

Resultados do Programa de Bolsas de Pesquisa (PBP-Loggi)

08.Dezembro.2021

I Workshop PBP-Loggi 2021

Resultados do Programa de Bolsas de Pesquisa (PBP-Loggi 2021)

Prefácio

A Loggi quer motivar e contribuir para o estabelecimento de um ecossistema de formação de recursos humanos, pesquisa e desenvolvimento, e inovação na cadeia logística, facilitando o estabelecimento de uma infraestrutura de entregas rápidas, com impacto econômico e social. A Loggi tem um time de tecnologia de alto nível e entende que explorar efetivamente as oportunidades de pesquisa, desenvolvimento e de formação de Recursos Humanos, conjuntamente, leva a um cenário mais produtivo, em termos de formação de recursos humanos e de produção de soluções envolvendo tecnologia de ponta.

O Programa de Bolsas de Pesquisa Loggi (PBP-Loggi) está inserido neste contexto, e almeja o estabelecimento de uma **rede de colaboração geograficamente distribuída** com as universidades. O PBP-Loggi - um programa piloto - foi estabelecido por meio de edital convite para as universidades que participaram do Roadshow Loggi entre 2019 e 2020.

Estudantes e orientadores de 15 universidades aplicaram e foram contemplados, com bolsas de IC e de Mestrado, em **todas as regiões do Brasil**. Os docentes orientadores e co-orientadores dessas propostas, alguns recém doutores, são docentes experientes e com produção científica e orientação de projetos pertinentes aos temas propostos.

O Workshop PBP-Loggi envolve a **apresentação de trabalhos de IC e de Mestrado** desenvolvidos, e visa caracterizar o estado atual dos trabalhos, resultados obtidos até o momento, trabalhos em andamento e perspectivas futuras. Observa-se que o PBP-Loggi motivou um conjunto de atividades de formação de recursos

humanos e de pesquisa em temas relacionados à cadeia logística, em particular da Loggi.

Além das apresentações dos trabalhos de IC e de Mestrado, o Workshop PBP-Loggi conta com 3 palestras:

- Palestra de Abertura
  - Visão Loggi
    Davi Reis (CTO/Loggi)
- Palestra Convidada
  - O Estado-da-Arte no CVRP
    Eduardo Uchoa (UFF)
- Palestra de Encerramento
  - Perspectivas Futuras em Roteirização e Malha Logística
    Gabriela Surita (Gerente de Engenharia/Loggi)

Esperamos que este workshop constitua um contexto integrador e indutor de uma **rede de colaboração na Cadeia Logística**, com diversidade geográfica, fortalecendo a relação Loggi-Academia. A Loggi sente-se muito honrada em ter como parceiros entidades e pesquisadores renomados, nacional e internacionalmente, e espera motivar e contribuir para a consolidação de um ecossistema na cadeia logística, domínio de aplicação com forte impacto econômico e social, promovendo a democratização dos serviços neste contexto.

Equipe Loggi:

Dhanielly Lima Fillipe Goulart Gabriela Surita Jose Carlos Maldonado
Juan Camilo Mayra Albuquerque Regiane Nascimento

Coordenação

Gabriela Surita

Regiane Nascimento

Mayra Albuquerque

Dhanielly Lima

Fillipe Goulart

José Carlos Maldonado

IES Participantes

UNIVERSIDADE FEDERAL DO AMAZONAS

UNIVERSIDADE FEDERAL DE ALAGOAS

UFRN UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

UFLA UNIVERSIDADE FEDERAL DE LAVRAS

UFOP UNIVERSIDADE FEDERAL DE OURO PRETO

EACH | campus capital USP LESTE
Escola de Artes, Ciências e Humanidades
Universidade de São Paulo

ufscar

UNIFESP
UNIVERSIDADE FEDERAL DE SÃO PAULO
1933

UFG UNIVERSIDADE FEDERAL DE GOIÁS

UNIVERSIDADE FEDERAL DO PARÁ

uff Universidade Federal Fluminense

UFMS UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL

UEFS UNIVERSIDADE ESTADUAL DE FEIRA DE SANTANA

UFMA Universidade Federal do Maranhão

FURG Fundação Universidade Federal do Rio Grande

Palestrantes

EDUARDO UCHOA

Eduardo Uchoa é professor do departamento de engenharia de produção da Universidade Federal Fluminense. Trabalha com Otimização, com especialidade em Programação Inteira. É um dos maiores especialistas do mundo na técnica que combina geração de colunas com planos de corte, nos chamados algoritmos de Branch-Cut-and-Price, já tendo contribuído com importantes inovações metodológicas. É um dos criadores do VRPSolver, que atualmente é o melhor algoritmo exato para a maioria das variantes clássicas do problema de roteamento de veículos. É o chair dos tracks CVRP e VRPTW do DIMACS 12th Implementation Challenge

GABRIELA SURITA

Gabriela Surita é formada em Engenharia de Computação pela Universidade Estadual de Campinas (Unicamp) em 2018 e é mestranda no programa de Engenharia Elétrica com ênfase em Computação pela mesma universidade. Também atua como Gerente de Engenharia na Loggi, coordenando as equipes de desenvolvimento centradas em soluções de roteirização e desenho de malha logística. Sua atividade de pesquisa é centrada em métodos baseados em aprendizado de máquina para a solução de problemas de otimização combinatória.

Engenharia

DAVI REIS

Cientista da Computação, começou sua carreira trabalhando como desenvolvedor e hacker de chapéu branco no mundo dos ISPs. Durante seu mestrado, trabalhou como um banco de dados e cientista de recuperação de informação, criou o RTDM algoritmo, foi coautor da biblioteca de código aberto CMPH e se juntou à Akwan, a startup que se tornou o Google Brasil. Trabalhou dez anos no Google, liderando projetos em Search, Mobile e Social Ads. Levou o Orkut de receita zero para lucratividade. Conduziu o projeto Google Pigeon, melhorando a busca local para bilhões de pessoas. Co-fundou, quase dimensionou e adquiriu o WorldSense. Nos últimos três anos, ele ajudou a Loggi a reconstruir a logística do país, recrutando uma equipe incrível de mais de 200 engenheiros e escalando o negócio 30x, criando uma das maiores empresas de entrega expressa do país.

Convidado

Alexandre Motta
Coordenação de Apoio à Pesquisa, Desenvolvimento e Aplicações do Conselho Nacional de Desenvolvimento Científico e Tecnológico

Trabalhos

I. Trabalhos de Mestrado

1. *Uma Abordagem Multidimensional para Roteirização no Território Inteligente*.....12
Autores: Matthieu Oliveira, Victor Ribeiro, Thais Batista, Everton Cavalcante e, Marcel Oliveira.
Afiliação Universidade Federal do Rio Grande do Norte - UFRN.

2. *Utilização de Machine Learning para Otimizar Problemas de Roteamento de Veículos na entrega Last Mile*.....18
Autores: Patricia Almeida da Silva e Regina Barwaldt.
Afiliação: Universidade Federal do Rio Grande - FURG.

***3. Comparação entre Operadores de Cruzamento para Algoritmo Genético ao Problema do Roteamento de Veículos com Restrição de Capacidade: Resultados Preliminares*.....24**
Autores: Ronaldd Pinho e Filipe Saraiva.
Afiliação: Universidade Federal do Pará - UFPA.

***4. Uma heurística baseada em Iterated Local Search para o Problema de Roteamento de Veículos com Entregas Fracionárias*.....28**
Autores: Alexandre dos Santos Mello, Luiz Satoru Ochi, Matheus Nohra Haddad e Igor Machado Coelho.
Afiliação: Universidade Federal Fluminense

***5 - Resolvendo instâncias de entregas urbanas reais com janelas de tempo baseadas no benchmark loggiBUD e aplicando um Branch-Cut-and-Price via VRPsolver*.....34**
Autores: Juan Rosas, Thailsson Andrade e Rosiane de Freitas
Afiliação: Universidade Federal do Amazonas.

***6 - Análise, Definição e Experimentação de Critérios de Teste de Software para Integração de Sistemas via API, Serviços ou Microsserviços*.....40**
Autores: Igor Ribeiro Nascimento Torga e Auri Marcelo Rizzo Vincenzi
Afiliação: Universidade Federal de São Carlos - UFSCAR.

7 - Otimização da satisfação do cliente na entrega de pacotes online utilizando demandas estocásticas.....46
Autores: Beatriz Cabral Santalucia e Alexandre da Silva Freire
Afiliação: Universidade de São Paulo - USP

8 - Um Problema Integrado de Planejamento da Produção e da Distribuição de Produtos Manufaturados.....51
Autores: Sasha K. D. A. B. H. Barros e Willy Alves de Oliveira Soler
Afiliação: Universidade Federal do Mato Grosso do Sul - UFMS.

II. Trabalhos de Iniciação Científica

Sessão 1: Roteirização Clássica

1 - Roteirização no território inteligente.....55
Autores: Matthieu Oliveira, Victor Ribeiro, Thais Batista e Everton Cavalcante e Marcel Oliveira
Afiliação: Universidade Federal do Rio Grande do Norte - UFRN.

2 - Seleção de Rotas Modelos para resolver o Last-Mile Incremental Capacitated Vehicle Routing Problem (ICVRP).....58
Autores: Ascânio Sávio de Araujo Neves, Ruan Heleno Correa da Silva, Bruno Costa e Silva Nogueira e Rian Gabriel Santos Pinheiro
Afiliação: Universidade Federal de Alagoas - UFAL.

3 - Resilient Routing.....64
Autores: Arthur Souza Ribeiro, Leonardo Bacelar Lima Santos e Lívia Rodrigues Tomás
Afiliação: Universidade Federal de São Paulo - UNIFESP e Cemaden.

4 - Uma abordagem baseada em clusterização hierárquica para resolver o Last-Mile Incremental Capacitated Vehicle Routing Problem (ICVRP).....69
Autores: Ruan Heleno Correa da Silva, Ascânio Sávio de Araujo Neves, Rian Gabriel Santos Pinheiro e Bruno Costa e Silva Nogueira
Afiliação: Universidade Federal de Alagoas - UFAL.

5 - Solução para o Problema de Roteamento de Veículos com Janela de Tempo Aplicado aos Conjuntos de Entradas do Benchmark Loggi BUD.....74
Autores: Élani Marques Zanlucas e Rúbia Mara de Oliveira Santos, Afiliação: Faculdade de Computação - FACOM e Universidade Federal de Mato Grosso do Sul - UFMS.

6 - Last-Mile Capacitated Vehicle Routing Problem.....78
Autores: Vinicius Gabriel Angelozzi Verona de Resende, Puca Huachi Vaz Penna e Túlio Ângelo Machado Toffolo
Afiliação: Universidade Federal de Ouro Preto - UFOP.

7 - Busca Tabu aplicada a resolução de uma variante do problema roteamento dinâmico de veículos.....84
Autores: Miguel Rodrigues Guimarães de Oliveira, Arthur Henrique Sousa Cruz, Franklina Maria Bragion de Toledo e Mayron Cesar de Oliveira Moreira
Afiliação: Universidade Federal de Lavras - UFLA e Universidade de São Carlos - UFSCar.

Sessão 2: Meta-heurísticas

8 - Meta-heurística GRASP aplicada ao problema do roteamento de veículos capacitado: um estudo de caso com a empresa Loggi.....90
Autores: Mayrla Santos Jansen e Glaubos Clímaco
Afiliação: Universidade Federal do Maranhão - UFMA.

9 - Estudo de Caso da Meta-heurística GRASP Aplicada ao Problema de Alocação de Hubs Capacitados.....95
Autores: Nathasha Pinto e Glaubos Clímaco
Afiliação: Universidade Federal do Maranhão - UFMA.

10 - Uma meta-heurística adaptativa aplicada ao problema de roteamento de veículos capacitados e variantes.....99
Autores: Angelina Gomes Santos; Antônio Augusto Chaves e Cleder Marcos Schenekemberg
Afiliação: Universidade Federal de São Paulo São, José dos Campos - UNIFESP.

11 - Um problema de roteamento de veículos com frota heterogênea - Modelo de otimização e heurísticas.....105
Autores: Lee Jing Xuan e Willy Alves de Oliveira
Afiliação: Universidade Federal de Mato Grosso do Sul - UFMS.

12 - Otimização por Meta-Heurísticas mediado por Sistemas Multiagentes aplicados ao Problema do Roteamento de Veículos Last-Mile.....111
Autores: Johann Pinheiro Pires e Eder Mateus Nunes Gonçalves
Afiliação: Universidade Federal do Rio Grande - FURG.

Sessão 3: Miscelanneous

13 - Processos Markovianos de Decisão: uma análise comparativa.....116
Autores: Alexia Carolina Scheffer da Silva e Valdinei Freire da Silva;
Afiliação: Universidade de São Paulo - USP.

14 - Loggi Hub Point (2021): Local ideal para a instalação de um novo Hub.....121
Autores: Juliana Aragão Pinto e João Batista da Rocha Junior
Afiliação: Universidade Estadual de Feira de Santana - UEFS.

15 - Habilitando mineração de processos no domínio de operações logísticas: extração de dados para um log de eventos centrado em objetos.....127
Autores: Mateus Alex dos Santos Luna, Elio Ribeiro Faria Junior, Thais Rodrigues Neubauer, Marcelo Fantinato e Sarajane Marques Peres
Afiliação: Universidade de São Paulo - USP.

16 - Alto nível vs baixo nível: Uma abordagem C++ para o problema de otimização da plataforma loggibud.....133
Autores: Fellipe Souza Pessanha, Igor Machado Coelho, e Luiz Satoru Ochi
Afiliação: Universidade Federal Fluminense - UFF.

17 - Um estudo comparativo de ferramentas de apoio ao teste de mutação para Python.....139
Autores: Davi Lobo de Oliveira e Auri Marcelo Rizzo Vincenzi

Afiliação: Universidade Federal de São Carlos - UFSCar.

18 - Especificação e Verificação Formal em Aplicações de Logística.....145
Autores: Victor Ribeiro e Marcel Oliveira
Afiliação: Universidade Federal do Rio Grande do Norte - UFRN.

19 - Middleware Específico de Domínio para Aplicações e Serviços de Entregas Urbanas.....147
Autores: Marcos Vinícius de Souza Oliveira e Fábio Moreira Costa
Afiliação: Universidade Federal de Goiás - UFG.

Uma Abordagem Multidimensional para Roteirização no Território Inteligente

Matthieu Oliveira, Victor Ribeiro, Thais Batista, Everton Cavalcante, Marcel Oliveira
Universidade Federal do Rio Grande do Norte
Natal, Brasil
matthieuceo@gmail.com,cstavictor@gmail.com,thaisbatista@gmail.com,everton.cavalcante@ufrn.br,marcel@dimap.ufrn.br

RESUMO

Este trabalho propõe uma abordagem multidimensional para análise do problema de roteirização, de forma a se determinar a(s) melhor(es) rota(s) considerando dados relacionados a diferentes domínios de uma cidade. A estratégia proposta define (i) uma função de qualidade para cada uma das dimensões consideradas, de forma a avaliar a qualidade da rota, e (ii) uma função de utilidade que considera simultaneamente as diferentes dimensões, com ponderação para cada uma delas à escolha do tomador de decisão. A abordagem proposta foi implementada em uma plataforma georreferenciada para desenvolvimento de cidades inteligentes, a *Smart Geo Layers* - SGeoL, a qual permite integrar dados de diversos domínios de uma cidade. Como prova de conceito, a SGeoL é usado combinar de dados de roteirização com dados de segurança, ambos da cidade do Natal, Rio Grande do Norte, apontando as melhores rotas resultantes de acordo com esses critérios.

PALAVRAS-CHAVE

logística, roteirização, território inteligente, análise multidimensional, plataforma georreferenciada

1 INTRODUÇÃO

A área de logística envolve o planejamento, implementação e controle eficiente da coleta, armazenamento e distribuição de encomendas. Nesse contexto, um serviço de *roteirização* consiste em um problema de otimização combinatória para a definição de trajetos (rotas) exatas para o transporte de encomendas, tendo como resultado a melhor solução para o transportador. Esse processo de definição das melhores rotas vai além da tarefa de encontrar a rota mais rápida, uma vez que outras informações relevantes devem ser também consideradas, tais como ambiente, congestionamento e segurança do tráfego [11], de forma a garantir a eficácia, eficiência e segurança da operação de transporte.

Diversas aplicações de logística estão inseridas no contexto de *cidades inteligentes*, as quais usam Tecnologias da Informação e Comunicação (TICs) em prol da melhoria da qualidade de vida da população [2]. Em cidades inteligentes, uma variada gama de dados coletados por sensores, dispositivos e sistemas sobre diferentes aspectos da cidade são integrados de forma a permitir uma análise conjunta de dados do território. Isso leva à concretização do conceito de *território inteligente*, o qual se refere à capacidade de analisar, diagnosticar e tomar decisões levando em consideração, de forma integrada, características geográficas, sociais, econômicas e culturais do território [3]. Essa visão multidimensional, aplicada ao contexto de logística, possibilita que seja feita uma roteirização baseada na análise de diversas variáveis da cidade que podem ter influência sobre a escolha de uma rota. A título de exemplo, a ocorrência de crimes em uma determinada região da cidade (ou seja, informações relacionadas ao contexto de segurança pública) pode ser determinante no traçado de rotas de entregas por transportadoras, que são informações associadas ao domínio de logística.

No contexto de uma empresa de logística como a Loggi, possíveis rotas são calculadas para cada entrega considerando critérios diversos. Essa informação das possíveis rotas para cada entrega pode ser combinada com inúmeras outras informações do território inteligente, tais como informações sobre problemas de segurança na cidade, dados de trânsito, problemas nas vias da cidade como, por exemplo, alagamentos e obras, dentre outras. Tais informações estão tipicamente disponíveis em plataformas integradoras para desenvolvimento de serviços e aplicações para cidades inteligentes [10]. O propósito da análise combinada de informações das rotas geradas com dados adicionais fornecidos pela cidade é indicar, dentre as rotas previstas, qual(is) a(s) melhor(es) considerando como critérios dados do território inteligente.

Nessa perspectiva, o objetivo deste trabalho é propor e implementar uma abordagem multidimensional para roteirização que considera informações providas pelo território inteligente. Nessa abordagem, é feita uma correlação de dados de rotas, a exemplo da distância total destas, com dados de outros domínios da cidade que podem ter impacto na escolha da rota, tais como informações relativas a ocorrências de segurança. As informações referentes às rotas e às ocorrências de segurança são organizadas na forma de *camadas* no âmbito da *Smart Geo Layers* - SGeoL [9], uma plataforma georreferenciada de desenvolvimento de aplicações para cidades inteligentes. A SGeoL organiza e gerencia informações de uma cidade representando-as na forma de camadas, cada uma delas composta por um conjunto de elementos (entidades) que possuem atributos geoespaciais. Dessa forma, a SGeoL possibilita combinar diversos domínios de uma cidade que podem impactar na decisão de roteirização, tais como dados sobre problemas de segurança (visando a integridade das encomendas), dados sobre problemas nas vias da cidade, entre outros.

O restante deste texto está estruturado da seguinte forma. A Seção 2 discorre brevemente acerca de alguns trabalhos relacionados. A Seção 3 apresenta a abordagem proposta para avaliação multidimensional de rotas. A Seção 4 descreve a implementação da

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
Workshop Virtual Loggi, 8 de dezembro de 2021, On-line
© 2021 Copyright held by the owner/author(s).

abordagem proposta e sua integração com a plataforma SGeoL. A Seção 5 traz algumas considerações finais e elenca trabalhos futuros.

2 TRABALHOS RELACIONADOS

Korczak e Kijewska [5] realizaram uma revisão de literatura acerca de logística inteligente (*smart logistics*, em inglês) sob a ótica do contexto de cidades inteligentes. Os autores ressaltam que dados obtidos de diferentes fontes devem ser detalhados, mensuráveis e coletados em tempo real e deve haver a possibilidade de agregá-los para uma análise conjunta. Esse trabalho confirma, assim, a importância do tratamento das questões de logísticas com informações de outros domínios de cidades inteligentes.

Kubek e Więcek [6] propuseram uma estratégia para tomada de decisão multicamadas de diversos problemas de logística, em especial para transporte. A estratégia proposta é a base para melhor integração e cooperação entre as entidades de logística da cidade, sendo centrada na integração de dados e na inteligência ambiental e econômica, visando melhorar o transporte de cargas nas cidades e diminuir o impacto ambiental. Nesse trabalho, problemas de logística são tratados sob a perspectiva de multicamadas visando especificamente o transporte com sustentabilidade ambiental. De forma similar, a proposta do presente trabalho possui vistas à análise integrada de dados de forma multidimensional, com foco, entretanto, na definição da(s) melhor(es) rota(s) considerando informações de diversos domínios da cidade que podem impactar na roteirização.

Abella et al. [1] ressaltam que cidades inteligentes são elementos essenciais para lidar com alguns dos maiores desafios que a sociedade enfrenta, como superpopulação, transporte, poluição, sustentabilidade, segurança, saúde, e que os portais de cidades inteligentes oferecem uma grande quantidade de dados que podem ser usados por entidades privadas e públicas para criar serviços. O trabalho apresenta um modelo que demonstra como os dados divulgados em cidades inteligentes geram valor para os cidadãos e para a sociedade. O modelo inclui várias das dimensões que tornam os dados atrativos para reutilização, além de analisar mecanismos de criação de produtos e serviços inovadores, mostrando como produtos e serviços impactam a sociedade. A visão fornecida por esse trabalho corrobora a proposta do presente trabalho em utilizar dados de diferentes domínios da cidade em uma análise conjunta para determinação de melhores rotas no contexto da Loggi.

3 ABORDAGEM MULTIDIMENSIONAL PARA AVALIAÇÃO DE ROTAS

O objetivo da abordagem multidimensional proposta neste trabalho é possibilitar a determinação da(s) melhor(es) rota(s) considerando dados relacionados a diferentes domínios de uma cidade. Com isso, procura-se responder à seguinte questão: *Como avaliar quantitativamente uma rota considerando múltiplas dimensões, por exemplo, distância total, aspectos de segurança, etc.?*

A metodologia empregada pela abordagem proposta consiste em três elementos. O primeiro elemento diz respeito à *caracterização* das diferentes dimensões de acordo com as camadas e possíveis sobreposições entre elas. O segundo elemento é a definição de uma *função de qualidade* para avaliar uma rota considerando cada uma das dimensões. Por sua vez, o terceiro elemento diz respeito à definição de uma *função de utilidade* para avaliar uma rota ponderando-se, de forma livre, cada uma das dimensões.

Esta seção apresenta, utilizando um exemplo ilustrativo simples, a abordagem proposta considerando a combinação de dados referentes à distância total de uma rota e informações relativas a ocorrências de segurança no território, possibilitando, assim, se ter uma avaliação de tal rota frente a outras possíveis. Cabe destacar que a abordagem ora proposta é flexível no sentido que é capaz de incorporar outras dimensões de interesse.

3.1 Caracterização das dimensões

A caracterização das diferentes dimensões de acordo com as camadas e possíveis sobreposições entre elas consiste em: (i) definir a camada que representa as informações consideradas; (ii) definir a variável de interesse que será considerada; (iii) identificar o valor que expressa de forma quantitativa a variável de interesse, e; (iv) a qualificação desse valor, se positiva (quanto maior, melhor) ou negativa (quanto menor, melhor). A Tabela 1 apresenta a caracterização de informações relativas a rotas e ocorrências, as quais serão consideradas no exemplo em questão.

3.2 Avaliação da qualidade de rotas

A avaliação da qualidade de uma rota consiste na definição de uma função de qualidade para cada uma das dimensões consideradas. Para a dimensão referente a rotas, a qualidade de uma rota r, dada por $Q(r)$, é igual à distância total entre os pontos de origem e de destino da rota (i e j), como define a Equação 1:

$$Q(r) = d(i, j) \tag{1}$$

Por sua vez, para a dimensão referente a ocorrências de segurança, a qualidade de uma rota r, dada por $Q(r)$, é igual à média da soma da distância euclidiana das ocorrências com relação aos n pontos que compõem a rota, conforme definido pela Equação 2:

$$Q(r) = \frac{1}{n} \sum_{o \in O} \sum_{p \in P} \sqrt{(o_x - p_x)^2 + (o_y - p_y)^2} \tag{2}$$

sendo $O = \{o_1, o_2, \ldots, o_m\}$ o conjunto de ocorrências, cada uma delas representada por um ponto com coordenadas (o_x, o_y), e $P = \{p_1, p_2, \ldots, p_n\}$ o conjunto de pontos que compõem a rota, cada ponto com coordenadas (p_x, p_y).

3.3 Avaliação da utilidade de rotas

Uma vez computada a avaliação da qualidade de uma rota, é necessário definir uma função de utilidade que considere simultaneamente as diferentes dimensões, cada uma delas podendo ser ponderada livremente à escolha do tomador de decisão. No entanto, como as variáveis de interesse são expressas em diferentes unidades, é necessário primeiro normalizar a qualidade de cada rota de acordo com a sua respectiva qualificação, positiva ou negativa. A normalização da qualidade de uma rota resulta em um valor inteiro $Q_N(r) \in [0, 1]$.

Para uma qualificação positiva (quanto maior, melhor), como é o caso da distância da localização das ocorrências com relação a uma rota, a qualidade normalizada de uma rota r, dada por $Q_N(r)$, é calculada conforme a Equação 3:

Tabela 1: Exemplo de caracterização de dimensões.

Camada	Variável de interesse	Valor	Qualificação
Rotas	Rota – sequência de linhas, cada uma definida por um par de pontos (p_x, p_y)	Distância total	Negativa
Ocorrências	Ocorrência – localizada no espaço por um par de coordenadas (o_x, o_y)	Distância de uma ocorrência com relação aos pontos que definem uma rota	Positiva

$$Q_N(r) = \frac{Q(r) - Q_{min}}{Q_{max} - Q_{min}} \qquad (Q_{max} \neq Q_{min}) \tag{3}$$

Para uma qualificação negativa (quanto menor, melhor), como é o caso da distância total de uma rota, a qualidade normalizada de uma rota r, dada por $Q_N(r)$, é calculada conforme a Equação 4:

$$Q_N(r) = \frac{Q_{max} - Q(r)}{Q_{max} - Q_{min}} \qquad (Q_{max} \neq Q_{min}) \tag{4}$$

sendo $Q(r)$ o valor de avaliação da rota r e Q_{min} e Q_{max} respectivamente o menor e o maior valor de qualidade dentre todas as possíveis rotas. Além disso, se $Q_{min} = Q_{max}$, então $Q_N(r) = 1$.

Finalmente, a função de utilidade para avaliar uma rota r para as múltiplas dimensões e pesos a elas atribuídos considera as medidas normalizadas de qualidade $Q_{Ni}(r)$ das n dimensões. Essa função de utilidade $u(r)$ é expressa pela Equação 5:

$$u(r) = \sum_{i=1}^{n} Q_{Ni}(r) * w_i \tag{5}$$

sendo $W = (w_1, w_2, \ldots, w_n)$ um vetor de pesos $w_i \in [0, 1]$ para ponderar cada dimensão i, com $\sum_{i=1}^{n} w_i = 1$. Dessa forma, a rota que apresentar utilidade maximal será considerada a melhor.

3.4 Exemplo

Para fins ilustrativos, considere-se o grafo direcionado mostrado na Figura 1 representando as possíveis rotas entre os pontos A (origem) e F (destino). Nesse grafo, os nós representam os pontos que compõem as rotas, cada um com suas respectivas coordenadas (p_x, p_y), e as arestas representam os segmentos de reta que formam as rotas, cada uma rotulada com um valor numérico que expressa a distância entre dois pontos. É possível visualizar nesse grafo a existência de quatro possíveis rotas entre A e F, a saber, $R_1 : A \rightarrow B \rightarrow C \rightarrow D \rightarrow F$, $R_2 : A \rightarrow C \rightarrow D \rightarrow F$, $R_3 : A \rightarrow F$ e $R_4 : A \rightarrow E \rightarrow F$. O grafo também apresenta a localidade das ocorrências, cada uma com suas respectivas coordenadas (o_x, o_y).

No exemplo em questão, considerando ambas as dimensões, é possível verificar que a rota mais curta não é necessariamente a mais segura e vice-versa. Como mostra a Tabela 2, utilizando a Equação 1 e a Equação 2 para avaliar a qualidade de cada uma das quatro rotas, tem-se que a rota R_4 é a de melhor qualidade em termos de distância ($Q(R_4) = 14$), porém é a menos segura ($Q(R_4) = 6,3629$). Por outro lado, a rota R_1 é a de melhor qualidade em termos da distância com relação às ocorrências registradas ($Q(R_1) = 8,1301$), ainda que seja a mais longa ($Q(R_1) = 27$). Fica evidente, portanto, a necessidade de uma análise que considere simultaneamente ambas as dimensões, ao invés de uma única dimensão de forma isolada, a fim de que seja possível decidir qual é de fato a melhor rota.

Figura 1: Exemplo de grafo representando as possíveis rotas entre dois pontos e a localização de ocorrências.

Seguindo a abordagem proposta, a próxima etapa diz respeito a normalizar a qualidade computada para cada rota para cada dimensão. Dado que a dimensão referente às ocorrências possui qualificação positiva e a dimensão referente à distância possui qualificação negativa, a Equação 3 e a Equação 4 são respectivamente utilizadas para normalizar a qualidade de cada rota. A Tabela 2 mostra a qualidade normalizada de cada dimensão para cada rota avaliada. A melhor rota para uma dimensão possui qualidade normalizada $Q_N(r) = 1,0000$ e, a pior rota, $Q_N(r) = 0,0000$.

Finalmente, a Equação 5 é utilizada para computar a utilidade de cada rota, de acordo com pesos atribuídos a cada dimensão. Como mostra a Tabela 2, para um vetor de pesos $W_1 = (0.5, 0.5)$ que atribui igual importância a ambas as dimensões, tem-se que a rota $R_3 : A \rightarrow F$ apresenta-se como a de utilidade maximal, ainda que não seja necessariamente a rota mais curta (R_4) nem a mais segura (R_1). Quando dá-se preferência a rotas mais seguras utilizando, por exemplo, um vetor de pesos $W_2 = (0.4, 0.6)$, a rota $R_1 : A \rightarrow B \rightarrow C \rightarrow D \rightarrow F$ é que apresenta utilidade maximal, ainda que seja a rota mais longa.

4 IMPLEMENTAÇÃO

Esta seção descreve as atividades relacionadas à concretização da abordagem proposta neste trabalho. Tais atividades incluem:

(1) a geração de instância dos pontos de entrega para o Estado do Rio Grande do Norte, a fim de possibilitar a determinação de rotas de entrega;
(2) a representação da instância gerada no formato GeoJSON [4];
(3) a geração de rotas a partir dos pontos de entrega na instância gerada;
(4) a implementação da abordagem multidimensional propriamente dita para avaliação de rotas, conforme a descrição apresentada na Seção 3, e;
(5) a integração com a plataforma georreferenciada SGeoL considerando as camadas de rotas e de segurança.

Tabela 2: Exemplo de computação da qualidade e da utilidade de rotas considerando as dimensões de distância e de segurança.

Rota	Qualidade		Qualidade normalizada		Utilidade	
	Distância	Segurança	Distância	Segurança	$W_1 = (0.5, 0.5)$	$W_2 = (0.4, 0.6)$
R_1	27	8,1301	0,0000	1,0000	0,5000	0,7000
R_2	24	7,7406	0,2307	0,7796	0,5052	0,6149
R_3	15	7,1935	0,9230	0,4701	0,6965	0,6059
R_4	14	6,3629	1,0000	0,0000	0,5000	0,3000

Figura 2: Instância gerada a partir do Loggi-BUD com pontos de entrega para a região metropolitana de Natal-RN.

Figura 3: Representação de pontos de entrega na camada *Delivery Points* na plataforma SGeoL.

Geração de instância para o Estado do Rio Grande do Norte. A versão atual do *Loggi Benchmark for Urban Deliveries* - Loggi-BUD [7], da Loggi, contém um conjunto de dados de entrega apenas de três regiões metropolitanas brasileiras, Rio de Janeiro, Brasília e Belém. Para gerar uma instância com pontos de entrega para a região metropolitana de Natal, capital do Estado do Rio Grande do Norte, foi utilizado o *pipeline* proposto no prório Loggi-BUD, juntamente com o motor de roteamento *Open Source Routing Machine* - OSRM [8], resultando na visualização mostrada na Figura 2. A descrição do processo para criação dessa instância está disponível no endereço https://bit.ly/loggibud-rn.

Representação da instâncias gerada no formato GeoJSON. Para possibilitar a representação adequada de informações geográficas, um dos formatos de dados utilizados para representar camadas na plataforma SGeoL é o GeoJSON [4]. O formato GeoJSON é um

Figura 4: Representação de rotas na camada *Routes* na plataforma SGeoL.

padrão aberto proposto como uma extensão do popular formato JSON, ao qual adiciona elementos que podem ser localizados no espaço geográfico, como pontos (incluindo endereços e localizações), linhas (incluindo ruas e estradas) e polígonos (para delimitar áreas no território), bem como suas propriedades. Contudo, a instância gerada a partir do Loggi-BUD é representada no formato JSON, sendo necessária, portanto, uma conversão de um formato para outro. Essa conversão foi implementada na linguagem de programação JavaScript, possibilitando a plotagem dos dados de pontos de entrega na SGeoL como uma camada. A Figura 3 mostra a camada *Delivery Points* na SGeoL, a qual representa os pontos de entrega da instância gerada para a região metropolitana de Natal-RN.

Geração de rotas. A partir da instância com os pontos de entrega, representados no formato GeoJSON, é possível traçar rotas utilizando os serviços disponibilizados pelo OSRM, considerando como ponto de origem um *hub*. A interação com o OSRM é feita mediante a realização de uma requisição HTTP GET ao serviço `route`, o qual é capaz de determinar a rota mais rápida entre coordenadas[1]. Essa requisição retorna como resultado um objeto *polyline*, o qual representa um conjunto de segmentos de reta resultante da codificação da sequência de coordenadas que formam uma rota. A

[1] A operação `routes` possibilita, em tese, gerar múltiplas rotas alternativas entre coordenadas, através da opção `alternatives=true`. No entanto, a própria documentação do OSRM para essa operação não garante a geração de rotas alternativas, ainda que estas sejam requisitadas: http://project-osrm.org/docs/v5.24.0/api/#route-service.

Figura 5: Visualização da sobreposição das camadas de roteirização e de segurança sobre a SGeoL.

geração de múltiplas rotas é feita de forma recursiva considerando múltiplos veículos, considerando o tamanho da entrega e limitado à capacidade total do veículo, aos quais se referem respectivamente os atributos `size` e `vehicle_capacity` de uma instância definida pelo Loggi-BUD.

Implementação da abordagem multidimensional para avaliação de rotas. A implementação da abordagem proposta neste trabalho considerou, a princípio, duas camadas de informação, (i) a camada *Delivery Points* anteriormente gerada, relativa aos pontos de entrega que definem possíveis rotas, e (ii) uma camada, já existente, relativa a pontos de ocorrência de crimes em bairros da cidade do Natal. Essa implementação foi realizada também na linguagem de programação JavaScript, estando disponível no endereço https://projetos.imd.ufrn.br/logrouter/algorithmsmartlogrouter. Em trabalhos futuros, a análise multidimensional irá considerar dados de outras camadas.

Integração com a SGeoL. A SGeoL é uma plataforma que fornece um conjunto abrangente de funcionalidades para facilitar o desenvolvimento de aplicações para cidades inteligentes, integrando dados heterogêneos de diferentes domínios e posicionando-os sobre o território por meio de informações geográficas. Dentre os recursos da plataforma, o *Dashboard de Desenvolvimento* fornece uma interface Web para auxiliar desenvolvedores na criação de novas aplicações, camadas e entidades de contexto, bem como no gerenciamento do controle de acesso às camadas e importação de dados. Por sua vez, o *Dashboard de Visualização* fornece aos usuários uma interface Web para visualizar entidades de contexto sobre o espaço urbano da cidade, além de possibilitar a correlação de informações de diferentes camadas. Com vistas a fazer uso dessas funcionalidades de desenvolvimento e visualização providas pela SGeoL, as rotas geradas e avaliadas, representadas no formato GeoJSON, foram importadas via requisições HTTP POST para a plataforma, sendo representadas como uma nova camada, *Routes*. A Figura 4 mostra a camada *Routes* na SGeoL.

A Figura 5 mostra uma sobreposição entre três camadas, duas relacionadas a roteirização (pontos de entrega, representados por pontos azuis, e rotas, representadas por linhas pretas) e uma relacionada a segurança (localidades de ocorrências, representadas por números em círculos amarelos e verdes). Tal sobreposição possibilita ao usuário visualizar, por exemplo, o traçado das rotas e a distância destas com relação às localidades de registro de ocorrências de segurança. Cabe destacar que outras camadas porventura existentes poderiam ser ainda sobrepostas com essas três, o que poderia ensejar a incorporação de novas dimensões a serem consideradas no processo de análise das rotas, seguindo a abordagem proposta neste trabalho.

5 CONSIDERAÇÕES FINAIS

Esse trabalho apresentou uma abordagem multimensional para roteirização que considera informações providas pelo território inteligente. Tal abordagem correlaciona os dados de rotas com dados de outros domínios da cidade que podem ter impacto na escolha da rota. A abordagem proposta consiste em: (i) caracterizar as diferentes dimensões de acordo com os domínios escolhidos, (ii) definir uma função para avaliar a qualidade de uma rota considerando cada uma das dimensões, e; (iii) definir uma função de utilidade para avaliar uma rota ponderando-se, de forma livre, cada uma das dimensões. A implementação da abordagem foi realizada em uma plataforma georreferenciada para aplicações de cidades inteligentes, a SGeoL. A prova de conceito, realizada para a cidade do Natal, mostra os resultados da correlação entre dados de rotas com informações de segurança.

Em termos de trabalhos futuros, pretende-se aprimorar a integração da abordagem multidimensional proposta neste trabalho para avaliação de rotas com a plataforma SGeoL. A SGeoL provê uma ampla gama de recursos que possibilitam aos usuários visualizarem informações georreferenciadas em camadas e realizar sobreposições entre elas, bem como o suporte ao desenvolvimento de aplicações

sobre estas [9]. Assim sendo, a integração consistirá no desenvolvimento de uma aplicação sobre a SGeoL que possibilite aos usuários não apenas visualizarem as rotas geradas e avaliadas, como também possam configurar as ponderações dadas às dimensões consideradas de acordo com seus interesses.

Outra linha de trabalho futuro está relacionada à incorporação de outras dimensões do território inteligente para avaliação das rotas. Devido ao fato de a abordagem proposta neste trabalho ser flexível, considerar outras dimensões de interesse consistiria apenas em (i) caracterizar cada dimensão considerando as informações disponíveis nas camadas e (ii) definir uma função de avaliação para avaliar quantitativamente cada dimensão. Dessa forma, a incorporação de novas dimensões tem o potencial de prover maior riqueza na avaliação das rotas por considerar diversas informações do território inteligente que poderão ter influência sobre a roteirização.

REFERÊNCIAS

[1] Alberto Abella, Marta Ortiz-de-Urbina-Criado, and Carmen De-Pablos-Heredero. 2017. A model for the analysis of data-driven innovation and value generation in smart cities' ecosystems. *Cities* 64 (April 2017), 47–53. https://doi.org/10.1016/j.cities.2017.01.011

[2] Vito Albino, Umberto Berardi, and Rosa Maria Dangelico. 2015. Smart cities: definitions, dimensions, performance, and initiatives. *Journal of Urban Technology* 22, 1 (2015), 3–21. https://doi.org/10.1080/10630732.2014.942092

[3] Carlo Giovanella. 2014. Smart territory analytics: Toward a shared vision. In *Proceedings of 47th Scientific Meeting of the Italian Statistical Society*. CUEC, Italy.

[4] Internet Engineering Task Force. 2016. RFC 7946 - The GeoJSON Format. https://tools.ietf.org/html/rfc7946.

[5] Jerzy Korczak and Kinga Kijewska. 2019. Smart Logistics in the development of smart cities. *Transportation Research Procedia* 39 (2019), 201–2011. https://doi.org/10.1016/j.trpro.2019.06.022

[6] Daniel Kubek and Paweł Więcek. 2019. An integrated multi-layer decision-making framework in the Physical Internet concept for the city logistics. *Transportation Research Procedia* 39 (2019), 221–230. https://doi.org/10.1016/j.trpro.2019.06.024

[7] Loggi. 2021. LoggiBUD: Loggi Benchmark for Urban Deliveries. https://github.com/loggi/loggibud.

[8] Dennis Luxen and Christian Vetter. 2011. Real-time routing with OpenStreetMap data. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, USA, 513–516. https://doi.org/10.1145/2093973.2094062

[9] Jorge Pereira, Thais Batista, Everton Cavalcante, Arthur Souza, Frederico Lopes, and Nélio Cacho. 2022. A platform for integrating heterogeneous data and developing smart city applications. *Future Generation Computer Systems* 128 (March 2022), 552–566. https://doi.org/10.1016/j.future.2021.10.030

[10] Eduardo Felipe Zambom Santana, Ana Paula Chaves, Marco Aurelio Gerosa, Fabio Kon, and Dejan S. Milojičić. 2017. Software platforms for smart cities: Concepts, requirements, challenges, and a unified reference architecture. *Comput. Surveys* 50, 6 (Nov. 2017). https://doi.org/10.1145/3124391

[11] Eiichi Taniguchi, Russell G. Thompson, Tadashi Yamada, and Ron van Duin. 2001. *City logistics: Network modeling and intelligent transport systems*. Emerald Publishing, United Kingdom.

Utilização de *Machine Learning* para Otimizar Problemas de Roteamento de Veículos na entrega *Last Mile*

Patricia Almeida da Silva
Mestranda em Engenharia de Computação
Universidade Federal do Rio Grande
Rio Grande, RS, Brasil
patriciaalmeida@furg.br

Regina Barwaldt
Docente do Centro de Ciências Computacionais
Universidade Federal do Rio Grande
Rio Grande, RS, Brasil
reginabarwaldt@furg.br

RESUMO

Problemas de Roteamento de Veículos (VRP) têm sido estudados há mais de 50 anos a fim de encontrar soluções para minimizar distâncias percorridas nas demandas de entrega. Com o avanço tecnológico, surgiu a necessidade de soluções capazes de lidar com o grande volume de dados das transportadoras. Este projeto tem como objetivo desenvolver um modelo para otimizar o VRP *Last Mile,* utilizando *Machine Learning*, capaz de fornecer caminhos mínimos. A primeira etapa do trabalho consiste na Revisão Sistemática da Literatura baseada no método proposto por Denyer e Tranfield em 2009. A pesquisa foi realizada a partir da análise das publicações compreendidas no período de 2016 a 2021, nas bases de dados: *ACM Digital Library, IEEE Xplorer, Science Direct, Springer Link, Wiley Online Library* e Biblioteca Digital Brasileira de Teses e Dissertações. Como resultado emergiu-se uma amostra final de 104 trabalhos, os quais foram categorizados de acordo com as soluções apresentadas e os métodos aplicados. Os resultados indicam que heurísticas e metaheurísticas são os métodos mais utilizados na resolução do problema. Dando seguimento ao trabalho, será desenvolvido um modelo para otimização do VRP *Last Mile* baseado em linguagem de programação *Python*. Para avaliação das distâncias será utilizado o *OpenStreetMaps* do servidor *Open Source Routing Machine.* O modelo baseado em redes neurais recorrentes será treinado com os dados do *Benchmark Loggi BUD* e os testes e validação se darão por meio de experimentos. Espera-se como resultado um modelo capaz de otimizar o VRP *Last Mile.*

CCS CONCEPTS

• *Computing methodologies* • *Machine learning* • *Machine learning approaches* • *Neural networks*

PALAVRAS-CHAVE

Roteamento de veículos, *last mile, Machine Learning, Benchmark Loggi BUD*.

1 Introdução

O Problema de Roteamento de Veículos (PRV), conhecido na literatura como *Vehicle Routing Problem* (VRP), foi originalmente proposto no final da década de 50, por Dantzig e Ramser, e consiste em estabelecer os trajetos de custos mínimos a serem percorridos por veículos com capacidade determinada, que partem de um mesmo ponto de origem, e têm como objetivo atender a demanda de clientes em localizações geograficamente dispersas[7].

No Brasil, no ano de 2020, o *e-commerce* teve um crescimento de 73,88%, conforme índices do MCC-ENET[1]. O crescimento do *e-commerce* e a subsequente pressão sobre as operações logísticas, aliado à exigência do nível de serviço por parte dos clientes, geram uma demanda por soluções inovadoras no processo de entrega de mercadorias. Assim, a busca pela otimização dos processos, auxiliados por Inteligência Artificial, tem se tornado cada vez mais recorrente[21].

Embora, muitas pesquisas tenham sido realizadas em busca de soluções para o VRP, é necessário resolver algumas aplicações práticas, pois, a maioria das pesquisas existentes concentram-se nas propriedades analíticas de diferentes variantes e os métodos de solução correspondentes, vários modelos matemáticos foram desenvolvidos, no entanto, a maioria são limitados a teóricos ou estudos empíricos de pequena escala. A implementação desses modelos em aplicativos do mundo real ainda enfrenta desafios consideráveis [1].

Um dos agravantes dos VRPs atuais é a entrega *last mile*, que consiste no problema de roteamento de veículos da última milha, o último trecho de uma entrega de pacote empresa-consumidor até o destinatário final que deve receber a mercadoria em casa ou em um ponto de coleta [10].

Considerada um dos segmentos mais caros de todo o processo de distribuição de mercadorias, a logística da

[1] MCC-ENET - indicador de consumo do mercado online, conjunto de indicadores de vendas do mercado online brasileiro.

última milha, é também a menos eficiente. Esse custo, se deve principalmente, ao elevado índice de entregas com falhas, ocasionadas pela ausência do cliente no endereço indicado[14].

Considerando as circunstâncias mencionadas, buscamos desenvolver um modelo para otimizar o VRP *Last Mile* utilizando *Machine Learning*.

A Inteligência Artificial sistematiza e automatiza tarefas intelectuais, processa dados por meio de algoritmos que tendem a se aperfeiçoar pelo seu próprio funcionamento (*self learning*) e a propor hipóteses diagnósticas cada vez mais precisas [18].

O *Machine Learning* pode ser considerado uma área de IA, que objetiva o desenvolvimento de técnicas computacionais sobre o aprendizado, bem como a construção de sistemas capazes de adquirir conhecimento de forma automática. Um sistema de aprendizado é um programa de computador que toma decisões baseado em experiências acumuladas através da solução bem sucedida de problemas anteriores [20].

1.1 Objetivos

O objetivo desta pesquisa, é desenvolver um modelo para otimização do VRP *Last Mile*, utilizando *Machine Learning*, que permita classificar as entregas em rotas de forma simultânea, à medida que chegam nos centros de distribuição finais, bem como fornecer rotas com custos mínimos.

Esse trabalho constituir-se-á a partir da análise dos estudos recentes sobre o VRP, estudo da estrutura do *Dataset Benchmark Loggi BUD*, estudo e especificação das técnicas de *Machine Learning*, seleção dos algoritmos para cálculo de caminhos mínimos, especificação de ferramentas para geração de mapas e rotas, desenvolvimento de uma arquitetura de rede neural recorrente capaz de processar os dados e retornar o caminho que contenha a menor distância, teste e validação do modelo.

2 Referencial Teórico

2.1 *Vehicle Routing Problem*

O problema de roteamento de veículos visa rotear um conjunto de veículos homogêneos com capacidade limitada para atender a demanda de um conjunto de clientes. Tem como objetivo minimizar a distância total percorrida pelos veículos de forma que cada veículo inicie e termine sua rota em um determinado depósito, cada cliente deve ser visitado uma vez por um único veículo, e a demanda total de uma rota não deve exceder a capacidade do veículo[22].

O VRP é um importante problema de otimização combinatória no qual busca-se encontrar a configuração ideal de rotas para uma frota de veículos que deve realizar entregas a partir de uma origem para um determinado conjunto de clientes[5].

Problemas VRP são comprovadamente considerados problemas NP-difícil, embora existem algoritmos exatos capazes de encontrar soluções ótimas em um tempo razoável para instâncias relativamente pequenas, esses problemas são computacionalmente difíceis e requerem alto custo computacional [16].

Desde a sua primeira formulação, surgiram outras variantes do VRP, que apresentam algumas restrições adicionais ao problema, representando condições presentes nas operações logísticas.

Com o elevado crescimento do comércio eletrônico nos últimos anos, evidenciou-se o problema VRP L*ast Mile* ou entrega na última milha. Existem algumas definições concorrentes, mas um entendimento amplamente aceito afirma que todas as atividades de logística relacionadas à distribuição de encomendas *online*, para famílias de clientes particulares em áreas urbanas, são consideradas de última milha. Conforme essa definição, a última milha começa quando uma remessa chega a um ponto de partida em uma área urbana, como um depósito central por exemplo, após o transporte de longa distância, e termina quando a remessa chega com sucesso ao ponto de destino do cliente final [4].

A logística *last mile* pode ser definida como a última parte do trânsito nas cadeias de abastecimento, em que as remessas são entregues dos centros de entrega até a porta dos clientes; entrega porta a porta (D2D)[17].

2.2 Soluções VRP *Last Mile*

Pesquisas operacionais que buscam otimizar o VRP *Last Mile*, cresceram exponencialmente, existindo uma variedade de trabalhos com diferentes abordagens.

Os pesquisadores [Cai et al., 2018] empregaram o *Algoritmo* de Busca Local Iterada *(ILS - Iterated Local Search)*. O *algoritmo* proposto baseou-se em uma sequência de buscas locais intercaladas por perturbações. As perturbações foram baseadas na metaheurística de Busca Adaptativa de Grande Vizinhança (*ALNS - Adaptive Large Neighborhood Search*). Foram utilizadas 4 instâncias de teste, obtendo uma eficácia de 75% [6].

O *Algoritmo* da otimização da Colônia de Formigas *(ACO - Ant Colony Optimization Algorithm)* foi empregado por [Zhang e Lee, 2016] para resolver o VRP na última milha integrado às opções de entrega a domicílio e ponto de coleta compartilhado, obtendo uma economia em relação a distância a ser percorrida e duração da viagem de 20,87% e 33,99%, respectivamente [26].

[Jiang et. al., 2021], propuseram uma heurística de duas fases baseada em *VNS (Variable Neighborhood Search)* pesquisa de vizinhança, a primeira fase buscou construir uma solução viável, e a segunda fase destinou-se a otimizar a solução encontrada. O algoritmo foi avaliado em 54 instâncias divididas em dois conjuntos, cada um deles com 27 instâncias. O primeiro grupo com instâncias de pequeno porte com 51 a 101 nós, e o segundo de médio porte com 130 a 200 nós. O algoritmo proposto teve uma eficácia de 98,15%, tendo conseguido encontrar a melhor solução para 53 instâncias [15].

Uma abordagem de otimização baseada em um conjunto de dados reais que combina técnicas de aprendizado de máquina com otimização de roteamento de veículos capacitados, foi proposta por [Chu et al., 2021], visando enfrentar os desafios da entrega na última milha [8].
Neste trabalho, categorizamos as soluções VRP *Last Mile*, em: Métodos Exatos, Heurísticas, Metaheurísticas, Técnicas de Inteligência Artificial e Métodos híbridos.

2.2.1 Métodos Exatos

Métodos exatos são métodos matemáticos utilizados para resolver problemas de otimização, buscam a melhor solução para o problema, o que não significa que essa seja a solução mais satisfatória. São utilizados em problemas de pequena ordem, pois, o tempo de processamento computacional torna-se muito elevado em problemas de ordem maior. Como exemplo de métodos exatos, podemos citar: Método *Branch and Bound*, Método de Decomposição de *Benders*, *Branch-and-Cut* e entre outros [12].

2.2.2 Heurísticas

Heurísticas consistem em um conjunto de regras e métodos que conduzem à descoberta, à invenção e à resolução de problemas, assim como os métodos exatos, elas fazem uso de métodos matemáticos, no entanto, buscam a solução mais satisfatória [25].
As heurísticas mais utilizadas para resolver o problema VRP são classificadas em: Heurísticas Construtivas, Heurísticas de duas fases e Heurísticas de melhoramento [25].

2.2.3 Metaheurísticas

Metaheurísticas consiste na aplicação e combinação de heurísticas, em cada passo da resolução do problema são aplicadas heurísticas determinadas para tratar de um problema específico, buscando encontrar boas soluções essas heurísticas são modificadas e guiadas[11].
Segundo [Benevides, 2011], uma metaheurística é um conjunto de conceitos que podem ser usados para definir os métodos heurísticos que podem ser aplicados a um vasto conjunto de problemas diferentes. Uma metaheurística pode ser vista como um quadro geral de algoritmos que podem ser aplicados a diferentes problemas de otimização com relativamente poucas modificações para torná-los adaptados a um problema específico [3].

2.2.4 Métodos Híbridos

Os Métodos Híbridos fazem uso de mais de um método de solução, é a junção de dois ou mais métodos, que buscam encontrar uma solução otimizada.

2.2.5 Redes Neurais Recorrentes

As Redes Neurais Recorrentes (RNNs) são um tipo especial de Rede Neural Artificial (RNA), que apresentam laços de realimentação local e/ou global em sua estrutura.

Nas RNNs, a saída de um neurônio é utilizada como entrada de outro neurônio, de uma camada anterior ao do primeiro, sendo a mais indicada para problemas que envolvem processamento temporal. A estrutura de uma RNN pode ser observada na Figura 1.

Figura 1: RNN [Eliasy; Przychodzeń, 2020]

Segundo [Silva, 2019], redes neurais recorrentes são modelos com a capacidade de seletivamente passar informação através de passos de uma sequência, durante o processamento de dados sequenciais elemento a elemento. Dessa maneira, uma rede neural recorrente é capaz de modelar entrada e/ou saída consistindo de sequências de elementos que não são independentes entre si [24].

2.3 Algoritmo de *Dijkstra*

O Algoritmo de *Dijkstra* foi desenvolvido em 1959 pelo Cientista da Computação *Edsger Wybe Dijkstra*, é utilizado para determinar o caminho mínimo entre dois vértices [2].
O algoritmo de *Dijkstra* é o mais conhecido e mais empregado dos algoritmos para cálculo de caminho. Tem como objetivo definir um vértice como ponto inicial e calcular o caminho de custo mínimo para percorrer todo o grafo [9].
Na Figura 2, temos a estrutura de um Grafo de *Dijkstra*, com nós, vértices e pesos dos vértices.

Figura 2 : Grafo de Dijkstra [Alrifai, 2015]

Escolhido um nó como origem, este algoritmo encontra o caminho mínimo deste nó para todos os demais nós da rede, parte de uma estimativa inicial para o caminho mínimo e vai sucessivamente ajustando esta estimativa, considera que um nó estará fechado quando já tiver sido obtido um caminho mínimo do nó origem até ele [23].

2.4 Revisão Sistemática da Literatura

Este estudo foi desenvolvido com base no processo sistemático proposto por [*Denyer* e *Tranfield* 2009], que dividem a Revisão Sistemática da Literatura em 5 etapas: (1) Formulação da Questão de Pesquisa, (2) Localização

dos estudos, (3) Avaliação e seleção dos Estudos, (4) Análise e Síntese, e (5) Relato e Uso de Resultados.
Propomos uma questão de pesquisa e algumas subquestões, responsáveis por nortear o estudo:
"Quais os métodos e soluções utilizadas para otimizar o problema de roteamento de veículos nas entregas na última milha e quais os fatores limitadores dos métodos propostos?"
As subquestões estão apresentadas na tabela abaixo:

Tabela 1: Subquestões de Pesquisa

Subquestões de Pesquisa	
Q1	Quais os métodos propostos na literatura para resolver o problema de roteamento de veículos?
Q2	Quais os principais métodos para solução do problema de roteamento de veículos nas entregas de última milha?
Q3	Quais os principais limitadores dos métodos propostos?

Selecionamos dentre os repositórios eletrônicos de pesquisas científicas, aqueles mais relevantes e com relação com o tema pesquisado.
As bases de dados escolhidas consistem em repositórios nacionais e internacionais de Artigos de Periódicos, Artigos de Conferências, Teses de Doutorado e Dissertações de Mestrado. Os repositórios escolhidos são: *ACM Digital Library*[2], *IEEE Xplorer*[3], *Science Direct*[4], *Springer Link*[5], *Wiley Online Library*[6] e a Biblioteca Digital Brasileira de Teses e Dissertações[7].
Para realizar a busca nesses repositórios, usamos as palavras-chave: (*solution, vehicle routing problem, last mile*), e os sinonimos: (*application, optimization, vehicle routes problem, vehicle itinerary problem*) bem como para otimizar a busca, utilizamos os conectores lógicos (*AND,OR*).
Obtivemos a seguinte *string*: ***(((“solution” OR “optimization” OR “aplication”) AND (“vehicle routing problem” OR “vehicle routes problem” OR “vehicle itinerary problem”)) AND“last mile”).***
A busca abrange publicações em Língua Portuguesa e Língua Inglesa, no período compreendido entre os anos de 2016 a 2021. Somente serão incluídos nessa pesquisa, Artigos de Periódicos, Artigos de Conferências, Teses de Doutorado e Dissertações de Mestrado.
Aplicando a *string* de busca em cada um dos repositórios de dados selecionados, considerando-se os metadados: título, resumo, palavras-chave, bem como os critérios estabelecidos anteriormente, obtivemos os resultados constantes na Tabela 2.

[2] https://dl.acm.org/
[3] https://ieeexplore.ieee.org/Xplore/home.jsp
[4] https://www.sciencedirect.com/
[5] https://link.springer.com/
[6] https://onlinelibrary.wiley.com/
[7] https://bdtd.ibict.br/vufind/

Tabela 2: Resultado Inicial da busca nos repositórios

Repositórios	**Total de Estudos**
ACM Digital Library	12
IEEE Xplorer	30
Science Direct	371
Springer Link	491
Wiley Online Library	49
BDTD	85
TOTAL GERAL	**1038**

As pesquisas iniciais, sem o uso de filtros e critérios, nos retornaram um total de 1038 estudos, aplicamos alguns critérios de inclusão e exclusão, visando garantir a qualidade da pesquisa.

- Inclusão de artigos escritos em Língua Portuguesa e Inglesa;
- Exclusão de artigos sem autoria;
- Exclusão de artigos sem permissão de acesso integral ao conteúdo;
- Exclusão de artigos introdutórios para questões especiais;
- Exclusão de trabalhos em livros, capítulos de livros e *Workshops*;
- Exclusão de artigos duplicados em diferentes fontes de dados; e
- Exclusão de artigos que não possuem relação com o tema da pesquisa.

Após o refinamento da busca, contamos com um total de 104 estudos selecionados. A Base de Dados *Science Direct*, apresentou maior contribuição para a pesquisa com um total de 36 trabalhos. A relação de trabalhos pertencentes aos demais repositórios, pode ser observada na Tabela 3.

Tabela 3: Resultado da busca após critérios

Repositórios	**Total de Estudos**
ACM Digital Library	07
IEEE Xplorer	15
Science Direct	36
Springer Link	16
Wiley Online Library	07
BDTD	23
TOTAL GERAL	**104**

Todos os trabalhos selecionados, apresentavam resposta para a questão **Q1: "Quais os métodos propostos na literatura no período de 2016 a 2021, para resolver o problema de roteamento de veículos?"** Os métodos para soluções VRP apresentados nos estudos foram sintetizados em 5 categorias, sendo as seguintes: Métodos Exatos, Heurísticas, Metaheuríísticas, Inteligência Artificial e Métodos Híbridos.
Dos 104 estudos selecionados, 47% das soluções propostas utilizam Heurísticas e Metaheurísticas, 18% aplicam Métodos Exatos, apenas 7% utilizam Inteligência Artificial e 28% fazem uso de Métodos Híbridos. Esses resultados podem ser observados na Figura 3.

Figura 3: Métodos de solução VRP

Apenas 54 estudos contemplam a questão **Q2: "Quais os principais métodos para a solução do problema de roteamento de veículos aplicados nas entregas de última milha?"** Destes, os métodos mais utilizados são Metaheurística e Métodos Híbridos, cada um deles conta com 28%. Esses resultados podem ser observados na Figura 4.

Figura 4: Métodos de solução VRP *Last Mile*

Quanto à questão **Q3: "Quais os principais limitadores dos métodos propostos?"** A maioria dos trabalhos utilizados nesta pesquisa, cerca de 95%, aplicavam seus métodos de solução em pequenos conjuntos de dados, envolvendo poucas instâncias, obtendo nesses casos, bons índices de precisão e redução de custos. No entanto, os trabalhos em que foram utilizadas bases de dados maiores, os índices dos resultados foram menos significativos.

3 Metodologia

A pesquisa iniciou-se com a Revisão Sistemática da Literatura, baseada em uma abordagem metodológica Quali-Quanti. Posteriormente, passamos para o estudo da estrutura e a organização do *Dataset Benchmark Loggi BUD*, que será utilizado como base de testes.
Estudos foram realizados, visando selecionar um algoritmo para cálculo de caminhos mínimos, para auxiliar no cálculo das rotas.
As distâncias entre os pontos serão avaliadas por meio do *OpenStreetMaps*[8] do servidor *Open Source Routing Machine*[9] *(OSRM)*.
Estudamos algumas técnicas de *Machine Learning*, visando selecionar a mais adequada ao projeto, selecionamos o *Framework Pytorch*[10].
O desenvolvimento do modelo se dará por meio da Linguagem de Programação *Python*, e a validação através de experimentos.

3.1 Proposta da aplicação

A proposta deste trabalho, consiste no desenvolvimento de um modelo com Algoritmo de *Dijkstra* e Redes Neurais Recorrentes, que possa otimizar o VRP *Last Mile*. Por meio do conjunto de dados *Benchmark Loggi BUD,* iremos treinar a rede neural que será responsável por processar os dados e retornar os caminhos mínimos.

3.1.1 *Dataset*

O dataset utilizado no trabalho trata-se do *Benchmark Loggi BUD*, um conjunto de dados para problemas de grande escala, que simulam o desafio de entregas na última etapa da cadeia logística. O *Loggi BUD* em sua versão inicial fornece 90 instâncias de treino e 30 instâncias de avaliação para as cidades de Rio de Janeiro (RJ), Brasília (DF) e Belém (PA) [19].
Os dados desse *dataset* estão estruturados em dicionários de dados *Json*[11], organizados conforme as cidades citadas. Cada *jso*n, contém um nome, a região atendida, as coordenadas do ponto de origem (latitude e longitude), a capacidade dos veículos e os pontos de entrega, constituídos de um identificador e coordenadas do endereço.
Inicialmente, criamos um *docker* para executar o servidor *OSRM*, com o servidor executando, extraímos o conjunto de dados em um diretório *OSRM.* Realizamos a manipulação e análise dos dados por meio da biblioteca Pandas[12] e com o auxílio do *OpenStreetMaps*, convertemos as coordenadas em endereços completos e plotamos em mapas, buscando identificar as regiões atendidas.
O próximo passo, foi obter as distâncias de um ponto ao outro, para utilizarmos no grafo do Algoritmo de *Dijkstra*. As distâncias foram obtidas através de requisição HTTPS ao

[8] *OpenstreetMaps*: projeto de mapeamento colaborativo para criar um mapa livre e editável.
[9] *OSRM*: mecanismo de roteamento de alto desempenho para os caminhos mais curtos nas redes rodoviárias
[10] Pytorch: biblioteca de aprendizado de máquina de código aberto baseada na biblioteca *Torch*, usada para aplicativos como visão computacional e processamento de linguagem natural.
[11] *JSON*: acrônimo de *JavaScript Object Notation*, utilizado para troca de dados entre sistemas, possui formato leve e simples.
[12] Pandas: biblioteca para manipulação e análise de dados no *Python.*

servidor *OSRM*, enviando como parâmetros as coordenadas de origem e destino e o modo de transporte (*car, foot, bike*, etc). Como resposta da requisição recebemos diversas informações, como a distância entre os pontos, o tempo de duração da viagem, o nome e as coordenadas dos pontos.

3.1.2 Proposta de Rede Neural

O modelo será baseado em redes neurais recorrentes, capazes de otimizar o VRP *Last Mile* e retornar os caminhos mínimos. Os dados a serem processados será o histórico de entregas, que será utilizado para treinamento da rede.

Iremos utilizar o *Framework Pytorch*. O uso dessa ferramenta, justifica-se pelos seguintes motivos: é *Open Source,* permite tanto o uso da CPU quanto da GPU para o treinamento do modelo, possui alto desempenho, apresenta bons recursos de depuração, consegue trabalhar com grandes conjuntos de dados e gráficos computacionais dinâmicos.

O modelo será desenvolvido com a linguagem de programação *Python.*

4 Resultados Parciais

Compreende-se por meio da análise e síntese dos resultados da pesquisa, que os métodos mais utilizados para solucionar problemas de roteamento de veículos consistem em heurísticas e metaheurísticas. No que tange, a problemas de roteamento de veículos na última milha, os métodos mais utilizados são metaheurística e métodos híbridos.

A maioria das soluções que obtiveram melhores resultados e otimalidade utilizaram instâncias e conjunto de dados pequenos para os testes.

Percebeu-se que embora exista um número significativo de pesquisas voltadas para o Problema de Roteamento de Veículos na última milha, apenas 13% dessas utilizaram IA.

Pretende-se, futuramente, desenvolver um modelo de Rede Neural Recorrente para otimizar o VRP *Last Mile,* auxiliando no processo de tomada de decisão de rotas, possibilitando a classificação de encomendas à medida que chegam aos centros de distribuição das transportadoras, agilizando o processo de entregas, diminuindo custos e consequentemente aumentando a satisfação dos clientes.

AGRADECIMENTOS

Agradecemos a parceria e apoio de toda a equipe da empresa Loggi.

REFERÊNCIAS

[1] Bai, R. et. al. 2021. Analytics and machine learning in vehicle routing research. Cornell University, arXiv:2102.10012v1 [cs.LG] 19 Feb 2021.

[2] Barros, E.A.R.; Pamboukian, S.V.D.; Zamboni, L.C. 2007. Algoritmo de Dijkstra: Apoio didático e multidisciplinar na implementação, simulação e utilização computacional. *In* ICECE - International Conference on Engineering and Computer Education, SP, Brasil.

[3] Benevides, P. 2011. A*plicação de Heurísticas e Metaheurísticas para o Problema do Caixeiro Viajante em um problema real de Roteirização de Veículos. Universidade* Federal do Paraná, UFPR.

[4] Boysen, N. et. al. 2020. *Last-mile delivery concepts: a survey from an operational research perspective. In* OR Spectrum, v.43, pages 1-58.

[5] Borowski, M. et. al. 2020. *New hybrid quantum annealing algorithms for solving vehicle routing problems*. In *ICCS International Conference on Computational Science*, v.12142, n.1, 2020.

[6] Cai, Z.; Zhang, Z.; He, H. 2018. Solving the last mile delivery problem using iterated local search approach. *In 2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC).*

[7] Caric´, T.; Gold, H. 2008. *Vehicle Routing Problem. In Intech Open.*

[8] Chu, H.; Zhang, W.; Bai, P. et al. 2021. Data-driven optimization for last-mile delivery. *In Complex & Intelligent Systems.*

[9] Falci, M.L.F.; Jabour, F.C. 2015. Um algoritmo de otimização de compras com suporte a rotas. *In* Revista Seminários de Trabalho de Conclusão de Curso do Bacharelado em Sistemas da Informação, Instituto Federal do Sudeste de Minas Gerais, MG, Brasil.

[10] Gevaers, R.; Voorde, E,V.T. 2009. *Characteristics of Innovations in last mile Logistics - using best practices, case studies and making the link with green and sustainable logistics. In* Association for European Transport and contributors.

[11] Glover, F.; Kochenberger, G. 2003. Handbook of metaheuristics. *In* 1. ed. Nova Iorque: Kluwer Academic Publishers

[12]Góes, A. R. T. 2005. Otimização na distribuição de carga horária de professores - Método exato, método heurístico, método misto e interface. *In* Universidade Federal do Paraná, UFPR, Curitiba, PR.

[13] Gomes, D. T. 2010. *Redes Neurais Recorrentes para previsão de séries temporais de memórias curta e longa. In* UNICAMP, SP.

[14] Ilhe, G. 2018. Last-Mile: o que é e o quanto isso impacta em sua operação? In E-commerce Brasil.

[15] Jiang, L.; Zang, X.; Dong, J. et. al. 2016. A covering traveling salesman problem with proft in the last mile delivery. *In Optimization Letters.*

[16] Kim, G. et. al. 2015. *City Vehicle Routing Problem (City VRP): A Review. In* IEEE Transactions on Intelligent Transportation Systems, v.16.

[17] Lim, S. et. al. 2016. *Three-Dimensional Vehicle Routing Problem for Urban Last Mile Logistics: Problem Formulation and Computational Analysis. In* 2016 UKSim-AMSS 18th International Conference on Computer Modelling and Simulation (UKSim), pages 252-257.

[18] Lobo, L. 2017. *Artificial Intelligence and Medicine. In* Revista Brasileira de Educação Médica, v.41, n.2, Rio de Janeiro, Brasil.

[19] Loggi. 2021. LoggiBUD: Loggi Benchmark for Urban Deliveries. *In GitHub repository. https://github.com/loggi/loggibud.*

[20] Monard, M.C.; Baranauskas, J.A. 2003. *Conceitos sobre Aprendizado de Máquina. In* Sistemas Inteligentes - Fundamentos e Aplicações. 1 ed.Barueri - SP: Editora Manole Ltda, pages 89-114.

[21] Pantarotto, C. 2018. *Otimização do roteamento e programação de veículo e drone aplicada a last mile delivery. In Universidade de São Paulo, Escola Politécnica, Departamento de Engenharia da Produção, SP, Brasil.*

[22] Salavati-Khoshghalb et. al.. 2019. *A hybrid recourse policy for the vehicle routing problem with stochastic demands. In* EURO Journal on Transportation and Logistics, v.8, n.1, pages 269-298.

[23] Silva, D. F.; Sanches, A.L. 2009. Aplicação Conjunta do Método de Dijkstra e Otimização Combinatória para Solução do Problema do Caixeiro Viajante. *In* SEGeT – Simpósio de Excelência em Gestão e Tecnologia.

[24] Silva, E.F. 2019. *Alocação das reservas internacionais com base em previsão de fatores econômicos e financeiros utilizando redes neurais recorrentes e modelo Nelson-Siegel Dinâmico. In* Universidade de Brasília, Instituto de Ciências Exatas, Departamento de Ciência da Computação.

[25]Wilhelm, V. 2013. Problema de roteamento de veículos. *In* Departamento de Engenharia de Produção – UFPR, PR.

[26] Zhang, S. Z.; Lee, C. K. M. 2016. Flexible vehicle scheduling for Urban Last Mile Logistics: The emerging technology of Shared Reception Box. *In 2016 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM).*

Comparação entre Operadores de Cruzamento para Algoritmo Genético ao Problema do Roteamento de Veículos com Restrição de Capacidade: Resultados Preliminares

Ronaldd Pinho
ronaldd.pinho@castanhal.ufpa.br
Universidade Federal do Pará
Instituto de Ciências Exatas e Naturais
Belém, PA, Brasil

Filipe Saraiva
saraiva@ufpa.br
Universidade Federal do Pará
Instituto de Ciências Exatas e Naturais
Belém, PA, Brasil

RESUMO

O problema de roteamento de veículos é um problema de classe NP-Difícil bem conhecido no campo da pesquisa em otimização combinatória e de grande interesse de muitos pesquisadores e da indústria. Pode-se dizer que é um típico problema em que há grande utilidade em usar metaheurísticas para buscar boas soluções. Nesse contexto, este trabalho visa apresentar a aplicação de uma técnica bem conhecida na literatura da área, o algoritmo genético. Com isso, são apresentados alguns estudos empíricos em relação à operação de cruzamento, que é considerada um componente essencial para o processo de busca dos algoritmos genéticos. Alguns resultados estatísticos de comparações entre os operadores *order crossover* (OX) e o *crossover parcialmente mapeado* (PMX) são apresentados neste trabalho.

KEYWORDS

algoritmo genético,crossover,problema de roteamento,otimização combinatória

ACM Reference Format:
Ronaldd Pinho and Filipe Saraiva. 2021. Comparação entre Operadores de Cruzamento para Algoritmo Genético ao Problema do Roteamento de Veículos com Restrição de Capacidade: Resultados Preliminares. In *Proceedings of Workshop Virtual Loggi: Resultados do Programa de Bolsas de Pesquisa (PBP Loggi).* ACM, New York, NY, USA, 4 pages.

1 INTRODUÇÃO

Na indústria de logística, empresas transportadoras de vários setores (naval, aéreo, rodoviário, etc) diariamente enfrentam um problema denominado por pesquisadores da área de pesquisa operacional como o "Problema do roteamento de veículos" (*Vehicle Routing Problem - VRP*) [5]. Esse é um problema bem conhecido e estudado devido a sua grande importância para a indústria e sua complexidade e variações, sendo uma das mais conhecidas o problema do roteamento de veículos com restrição de capacidade (*capacitated vehicle routing problem* - CVRP) [13] que adiciona uma restrição ao problema citado anteriormente e tem sido muito estudado nos últimos anos.

No geral, o CVRP é considerado como uma união de dois outros problemas bastante conhecidos: o problema do roteamento de veículos e o problema do empacotamento. O CVRP consiste em minimizar a distância percorrida por um conjunto de veículos partindo de um mesmo armazém a fim de visitar um conjunto de pontos de entrega onde cada entrega tem um peso e o peso total para um veículo entregar não pode ultrapassar a sua capacidade máxima.

Por se tratar de um problema NP-Difícil, é muito comum o uso de algoritmos heurísticos para buscar soluções semi-ótimas para instâncias consideravelmente grandes do problema. Muitos autores já aplicaram com sucesso diversos algoritmos conhecidos na literatura para resolver o problema do roteamento de veículos com restrição de capacidade, como por exemplo algoritmo de colônia de abelhas [12], algoritmos genéticos [1, 6, 10, 11], colônia de formigas [8] e muitas outras técnicas metaheurísticas.

Esse artigo se insere nesse contexto de aplicações de metaheurísticas ao CVRP, consistindo em aplicar uma das técnicas de otimização mais famosas, o algoritmo genético (AG), e com isso efetuar estudos em torno de um dos principais componentes dessa técnica, o operador de cruzamento, que mistura informações de duas soluções para obter novas e é considerado por muitos pesquisadores da área como a principal operação que implementa a busca nos AGs. Com isso, esse trabalho aplica um estudo comparativo entre o operador de cruzamento conhecido como *Order Crossover* (OX) e o operador de cruzamento parcialmente mapeado (PMX - *Partially Mapped Crossover*) afim de obter informações sobre a opção com melhor desempenho para o CVRP.

O artigo está organizado da seguinte forma: a seção 2 fala mais sobre a definição do problema do roteamento de veículos com restrição de capacidade; a seção 3 explica sobre as decisões referentes ao algoritmo genético, a codificação usada e os recursos utilizados no trabalho de implementação; a seção 4 explica a modelagem dos experimentos bem como apresenta os resultados preliminares obtidos com os mesmos; por fim na seção 5 são discutidas as ideias dos autores para a continuidade do presente trabalho de pesquisa.

2 MODELAGEM DO PROBLEMA

O problema de roteamento de veículos com restrição de capacidade ou mesmo roteamento de veículos capacitados é considerado uma conjunção de dois problemas clássicos de otimização: o Problema do Empacotamento e o Problema do roteamento de veículos.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
PBP Loggi, 8 de Dezembro de 2021, São Paulo, SP, Brasil
© 2021 Association for Computing Machinery.

Figura 1: Demonstração visual do problema de roteamento de veículos com 5 veículos. Autores: Paulo Funes e Thiago Poiani (CC-BY 4.0). Disponível em https://vehicle-routing-problem.github.io/img/vrp.png

O problema pode ser descrito da seguinte forma: dado um grafo $G = (V, E)$ onde o conjunto de vértices V é composto por pontos de entrega e um único depósito, e o conjunto de arestas E tal que $(v, u) \in E$ sendo $v \in V$ e $u \in V$, e toda aresta possuindo um peso (distância) não negativa assim como os vértices (peso da entrega), com exceção do vértice depósito, o objetivo é definir para um conjunto de M veículos que saem do depósito rotas sem intersecções entre si que tenham a menor distância total possível levando em consideração que a capacidade máxima de cada veículo não pode ser ultrapassada.

Trata-se de um problema de minimização, onde o objetivo é minimizar a soma da distância de todas as rotas percorridas, uma para cada veículo, atendendo a uma restrição de capacidade máxima que um veículo pode entregar. Um exemplo de solução em um problema de roteamento de tamanho pequeno pode ser visualizado na Figura 1.

3 ALGORITMO GENÉTICO

Os algoritmos genéticos são algoritmos de otimização adaptativos e baseados em população que utilizam os conceitos da teoria da evolução e seleção natural [4]. De maneira bem generalista, eles agem sobre uma população de soluções efetuando operações como cruzamento, que consiste em misturar duas ou mais soluções, e mutação, consistindo em fazer uma alteração mínima e aleatória em uma solução, imitando assim um processo de evolução em busca de tornar a população (conjunto de soluções) cada vez melhor dada uma função de avaliação (função objetivo) em tempo hábil.

Muitas implementações e variações dos AGs já foram propostas ao longo do tempo. Nesse trabalho, optou-se por utilizar como base a versão precursora da técnica que é conhecida como "algoritmo genético simples" (ou SGA, da sigla em inglês) com elitismo. Esse algoritmo foi escolhido por consistir somente das etapas essenciais de um algoritmo genético, o que o torna uma boa escolha para estudos focados em uma das operações básica dessa metaheurística. Além disso essa versão representa o fluxo original do AG definido inicialmente por Goldberg no fim dos anos 80 [4].

3.1 Codificação

A codificação de soluções - também chamadas de cromossomos - é uma etapa crucial dos AGs, pois ela pode inclusive impactar a performance da busca do algoritmo [6]. O problema do roteamento de veículos possui muita similaridade com outro problema bastante estudado na literatura, o problema do caixeiro viajante, e é bastante comum se utilizar o mesmo esquema de permutações como codificação. Nesse trabalho foi utilizado tal esquema que consiste em atribuir a cada uma das N entregas um valor inteiro único e dispor os valores em uma permutação simples.

O processo de decodificação utilizado foi escolhido a fim de garantir tanto a variação no número de veículos quanto o atendimento à restrição de capacidade máxima, dessa forma as operações de variação como cruzamento e mutação não impactam na viabilidade das soluções. O procedimento consiste em adicionar sequencialmente as entregas em uma rota até que a capacidade máxima seja alcançada, quando isso acontece a rota atual é fechada e a próxima entrega vai para uma nova rota, e o procedimento continua até que todas as entregas estejam em alguma rota.

Suponha uma instância do CVRP com 10 entregas e a capacidade máxima de carga dos veículos $C = 50$, o exemplo a seguir mostra o resultado de uma decodificação de solução usando como base as informações de pesos ilustradas na Tabela 1, dada uma solução codificada como

$$\{3, 8, 1, 4, 2, 5, 7, 6, 10, 9\}$$

após o processo de decodificação são definidas rotas considerando o valor 0 (zero) como sendo o ponto de partida e chegada dos veículos, ou seja, o depósito. As rotas geradas a partir dessa solução são:

Rota 1: $0 \longrightarrow 3 \longrightarrow 8 \longrightarrow 1 \longrightarrow 0$
Rota 2: $0 \longrightarrow 4 \longrightarrow 2 \longrightarrow 5 \longrightarrow 7 \longrightarrow 0$
Rota 3: $0 \longrightarrow 6 \longrightarrow 10 \longrightarrow 9 \longrightarrow 0$

Figura 2: Convergência média do algoritmo genético com o OX e com o PMX

Entrega	1	2	3	4	5	6	7	8	9	10
Peso	13	16	19	10	12	13	12	17	16	11

Tabela 1: Exemplo de distribuição de pesos entre as entregas de uma instância do CVRP.

3.2 Order Crossover (OX)

O operador de crossover baseado em ordem foi proposto por Davis [9] para o problema do caixeiro viajante, mas pode ser aplicado em qualquer problema que possa ter soluções representadas usando um esquema de codificação de permutações de números inteiros.

Dados dois cromossomos $p1$ e $p2$ esse operador define dois pontos de cortes aleatoriamente e todos os genes de cada cromossomo pai entre esses pontos são herdados pelos cromossomos filhos $f1$ e $f2$ nas mesmas posições, depois disso o operador faz para cada cromossomo filho uma busca nos genes do outro cromossomo pai, primeiro buscando pelos genes entre os pontos de corte que não estão na subcadeia herdada e depois faz a mesma busca fora dos pontos de corte, colocando-os nas posições de fora dos pontos de corte da esquerda para a direita na ordem em que os encontra nessas buscas. O exemplo a seguir ilustra o resultado da aplicação do operador OX.

$$p1 = \{\ 7\ 3\ 5 \mid 2\ 4\ 1 \mid 6\ 8\ \}\ \ \&\ \ p2 = \{\ 2\ 6\ 7 \mid 8\ 1\ 3 \mid 5\ 4\ \}$$
$$f1 = \{\ 8\ 3\ 6 \mid 2\ 4\ 1 \mid 7\ 5\ \}\ \ \&\ \ f2 = \{\ 2\ 4\ 7 \mid 8\ 1\ 3 \mid 5\ 6\ \}$$

3.3 Partially Mapped Crossover (PMX)

Esse operador de crossover foi proposto incialmente por Goldberg e Lingle [9] também para o problema do caixeiro viajante. De maneira similar ao OX ele cria dois pontos de corte aleatórios nos cromossomos pais p_1 e p_2, e então os trechos de genes entre os dois pontos definidos são chamados de "seções de mapeamento" e elas definem um relacionamento entre os cromossomos. Suponha a mesma dupla de cromossomos pais e pontos de corte aleatórios vistos anteriormente:

$$p_1 = \{\ 7\ 3\ 5 \mid 2\ 4\ 1 \mid 6\ 8\ \}$$
$$p_2 = \{\ 2\ 6\ 7 \mid 8\ 1\ 3 \mid 5\ 4\ \}$$

Então podemos considerar um mapeamento entre $2 \leftrightarrow 8$, $4 \leftrightarrow 1$ e $1 \leftrightarrow 3$. Após isso, a seção de mapeamento de primeiro cromossomo pai é herdada pelo segundo cromossomo filho e vice-versa:

$$f_1 = \{\ ?\ ?\ ?\ \mid 8\ 1\ 3 \mid ?\ ?\ ?\ \}$$
$$f_2 = \{\ ?\ ?\ ?\ \mid 2\ 4\ 1 \mid ?\ ?\ ?\ \}$$

Por fim, cada gene *i-ésimo* faltante em f_n é substituído pelo *i-ésimo* gene em p_n e caso o gene já esteja na seção de mapeamento de f_n ou mesmo já tenha sido adicionado, então é adicionado no lugar o gene mapeado por ele.

$$f_1 = \{\ 7\ 4\ 5 \mid 8\ 1\ 3 \mid 6\ 2\ \}$$
$$f_2 = \{\ 8\ 6\ 7 \mid 2\ 4\ 1 \mid 5\ 3\ \}$$

3.4 Implementação computacional

Foi utilizada a linguagem de programação C++ para as implementações computacionais, que foi escolhida principalmente devido sua performance. O algoritmo genético e todos os seus operadores foram implementados com o auxílio da biblioteca ParadisEO [2] - um *framework* para implementação de algoritmos metaheurísticos. Além disso, no que diz respeito à dados para representações do problema de roteamento de veículos, foi usado o LoggiBUD [7] como fonte de instâncias do CVRP para efetuar as experimentações.

4 EXPERIMENTOS E RESULTADOS PRELIMINARES

4.1 Modelagem dos experimentos e recursos usados

Os experimentos conduzidos consistem de testes computacionais com o algoritmo genético exaustivamente com cada um dos operadores de cruzamento abordados, portanto, foram realizadas 100 execuções do AG para a instância "cvrp-0-pa-100" com 298 entregas gerada através do LoggiBUD para cada operador, totalizando 200 execuções em que foram usadas a mesma combinação para os demais parâmetros do AG que podem ser vistos na Tabela 2.

Foi definido um critério de parada baseado no número de avaliações, ou seja, o número de vezes que a função objetivo foi calculada. Essa abordagem foi escolhida por ser comumente utilizada em concursos científicos de algoritmos de otimização.

As execuções foram efetuadas em um computador pessoal usando sistema operacional Manjaro com o kernel Linux na versão 5.13, processador Intel® Core™ i7-8665U Octa-core com 1,9 GHz e 16GB de memória RAM. Além disso, foi usada a versão 11.1 do GCC na compilação do código-fonte do projeto.

Parâmetro	Valor
Tamanho da população	100
Operador de Seleção	Torneio Determinístico
Operador de Mutação	Mutação Swap
Taxa de mutação	1%
Taxa de cruzamento	80%
Critério de parada	1000 (mil) avaliações

Tabela 2: Parâmetros fixados em todas as execuções do AG.

4.2 Resultados

Os estudos foram focados em dois dados principais relacionados ao desempenho em termos de obtenção das melhores soluções de cada um dos operadores de cruzamento abordados: (1) a convergência média do processo de evolução do algoritmo, que informa sobre o comportamento do processo de busca por soluções e (2) a média das soluções finais obtidas pelas execuções do AG.

Observações em torno das convergências e o comportamento geral da evolução considerando 100 execuções demonstram uma vantagem em se escolher o OX (order crossover) em detrimento do cruzamento parcialmente mapeado (PMX). Como pode ser observado na Figura 2 ambos operadores de cruzamento demonstram uma capacidade de variação da população semelhante dado os seus respectivos desvios padrão por geração, contudo, o operador OX teve um ganho considerável em eficácia na otimização da instância, chegando a minimizar a distância até valores próximos de 3600 quilômetros e com uma média um pouco acima de 3700 Km, enquanto o PMX obteve resultados cuja média não é menor que 4000 Km.

Mais informações sobre os resultados finais obtidos pelo AG com cada operador de cruzamento são apresentados na Tabela 3.

	OX	PMX
Melhor resultado	3333,06 Km	3774,25
Média	3716,34 Km	4051,22
Desvio padrão	162,06	119,65

Tabela 3: Melhores resultados e médias para cada operador.

5 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

No presente artigo foi apresentado um breve estudo em torno do uso de dois operadores de cruzamento conhecidos para o esquema de codificação inteira em algoritmos genéticos, o Order Crossover e o *Partially Mapped Crossover*, quando aplicado ao problema do roteamento de veículos com restrição de capacidade.

No estudo, esses operadores tiveram seus desempenhos em termos da qualidade das soluções encontradas comparados entre si. Os resultados obtidos demonstram certa vantagem do OX sobre o PMX.

Entretanto, cabe ressaltar que deve-se tomar algumas precauções ao se analisar os resultados desse estudo: por exemplo, uma única instância foi utilizada para testes, o que não é capaz de representar todas as escalas do CVRP. Por outro lado, outros operadores de cruzamento também não foram avaliados, como o *Linear Order Crossover* (LOX) e o *Cycle Crossover* (CX).

Como próximos passos na execução desse trabalho de pesquisa, os autores pretendem implementar um algoritmo de GRASP (*Greedy Randomized Adaptive Search Procedure*) [3] e efetuar estudos comparativos entre o mesmo e o atual algoritmo genético implementado como uma forma de comparar duas metaheurísticas de diferentes classes (uma populacional e outra de solução única) e, a partir dos resultados das comparações, propor uma hibridização entre algoritmo genético e GRASP. Além disso, também pretende-se implementar outros operadores de cruzamento comuns à codificações inteiras, e utilizar mais instâncias teste do LoggiBUD afim de representar um número maior de versões desse problema, com diferentes tamanhos e disposição de pontos de entregas no mapa, tendo assim uma maior variedade no resultados obtidos em estudos.

REFERÊNCIAS

[1] Barrie M. Baker and M.A. Ayechew. 2003. A genetic algorithm for the vehicle routing problem. *Computers & Operations Research* 30, 5 (April 2003), 787–800. https://doi.org/10.1016/S0305-0548(02)00051-5

[2] S. Cahon, N. Melab, and E.-G. Talbi. 2004. ParadisEO: A Framework for the Reusable Design of Parallel and Distributed Metaheuristics. *Journal of Heuristics* 10, 3 (May 2004), 357–380. https://doi.org/10.1023/B:HEUR.0000026900.92269.ec

[3] Thomas A. Feo and Mauricio G. C. Resende. 1995. Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization* 6, 2 (March 1995), 109–133. https://doi.org/10.1007/BF01096763

[4] David E. Goldberg. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning* (1st ed.). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[5] Gilbert Laporte and Ibrahim H. Osman. 1995. Routing problems: A bibliography. *Annals of Operations Research* 61, 1 (Dec. 1995), 227–262. https://doi.org/10.1007/BF02098290

[6] Stanley Jefferson de Araujo Lima and Sidnei Alves de Araújo. 2020. Genetic Algorithm applied to the Capacitated Vehicle Routing Problem: an analysis of the influence of different encoding schemes on the population behavior. *American Academic Scientific Research Journal for Engineering, Technology, and Sciences* 73, 1 (Oct. 2020), 96–110. https://asrjetsjournal.org/index.php/American_Scientific_Journal/article/view/6319

[7] Loggi. 2021. LoggiBUD: Loggi Benchmark for Urban Deliveries. https://github.com/loggi/loggibud. *GitHub repository* (2021).

[8] Silvia Mazzeo and Irene Loiseau. 2004. An Ant Colony Algorithm for the Capacitated Vehicle Routing. *Electronic Notes in Discrete Mathematics* 18 (2004), 181–186. https://doi.org/10.1016/j.endm.2004.06.029

[9] Zbigniew Michalewicz. 1996. *Genetic algorithms + data structures = evolution programs* (3rd rev. and extended ed ed.). Springer-Verlag, Berlin ; New York.

[10] Mazin Abed Mohammed, Mohd Sharifuddin Ahmad, and Salama A. Mostafa. 2012. Using Genetic Algorithm in implementing Capacitated Vehicle Routing Problem. In *2012 International Conference on Computer Information Science (ICCIS)*, Vol. 1. 257–262. https://doi.org/10.1109/ICCISci.2012.6297250

[11] Habibeh Nazif and Lai Soon Lee. 2012. Optimised crossover genetic algorithm for capacitated vehicle routing problem. *Applied Mathematical Modelling* 36, 5 (May 2012), 2110–2117. https://doi.org/10.1016/j.apm.2011.08.010

[12] W.Y. Szeto, Yongzhong Wu, and Sin C. Ho. 2011. An artificial bee colony algorithm for the capacitated vehicle routing problem. *European Journal of Operational Research* 215, 1 (2011), 126–135. https://doi.org/10.1016/j.ejor.2011.06.006

[13] Paolo Toth and Daniele Vigo. [n. d.]. *1. An Overview of Vehicle Routing Problems.* 1–26. https://doi.org/10.1137/1.9780898718515.ch1 arXiv:https://epubs.siam.org/doi/pdf/10.1137/1.9780898718515.ch1

Uma heurística baseada em *Iterated Local Search* para o Problema de Roteamento de Veículos com Entregas Fracionárias

ALEXANDRE DOS SANTOS MELLO*, Universidade Federal Fluminense, BR
LUIZ SATORU OCHI, Universidade Federal Fluminense, BR
MATHEUS NOHRA HADDAD, Universidade Federal de Viçosa, BR
IGOR MACHADO COELHO, Universidade Federal Fluminense, BR

A partir do problema clássico de roteamento de veículos (PRV) e com uma abordagem de um ou mais atendimentos de um cliente para suprir a demanda temos uma heurística baseada em *Iterated Local Search* para o Problema de Roteamento de Veículos com Entregas Fracionárias. Além da utilização do método de *Variable Neighborhood Descent* para as buscas locais, e ainda inúmeras estruturas de vizinhanças.

Additional Key Words and Phrases: Problema de Roteamento de Veículos, Entregas Fracionárias, Algortmo Heurístico, *Iterated Local Search*, *Variable Neighborhood Search*

ACM Reference Format:
Alexandre Dos Santos Mello, Luiz Satoru Ochi, Matheus Nohra Haddad, and Igor Machado Coelho. 2018. Uma heurística baseada em *Iterated Local Search* para o Problema de Roteamento de Veículos com Entregas Fracionárias . *ACM Trans. Graph.* 37, 4, Article 111 (August 2018), 6 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

O Problema de Roteamento de Veículos (PRV) surgiu a partir do *Truck Dispatcher Problem* em 1959 por Dantzing e Ramser [G. B. Dantzig and J. H. Ramser 208] com o objetivo de determinar rotas com o menor custo para a frota de caminhões para abastecer os postos. Na definição clássica do problema temos uma frota homogênea de veículos inicialmente posicionados na origem (depósito) e os clientes distribuídos, onde contém uma demanda e custos de deslocamento entre cada cliente no grafo. O PRV deve obedecer a três restrições:

- Toda rota inicia e termina na origem;
- A soma das demandas dos clientes atendidos por uma determinada rota não pode exceder a capacidade do veículo;
- Cada cliente tenha toda sua demanda suprida em uma única visita;

Logo o objetivo do PRV é determinar as rotas de cada veículo para atender todos os clientes minimizando o custo. O problema clássico com o passar do tempo foi adquirindo novas restrições gerando assim variantes do problema:

- Rotas com um ou múltiplos depósitos;
- Número de viagens por veículo;
- Janelas de tempo e limite de temo para concluir uma rota;
- Rotas limitadas ou ilimitadas
- Atendimento de demandas com uma ou mais visitas

A partir do problema clássico e levando em conta uma nova caraterística do atendimento com uma ou mais visitas, essa variante ficou conhecida na literatura como o Problema do Roteamento de Veículos com Entregas Fracionárias (PRVEF), relaxando assim a terceira restrição do problema clássico. Sendo considerado um problema da classe NP-difícil, encontrar a solução ótima pode demandar tempos computacionais proibitivos. Na indústria brasileira estima-se que 70% do valor final do produto [Olli Bräysy and Michel Gendreau 2005], logo estudos que conseguem otimizar o transporte são de grande interesse das organizações.

Com os estudos e uso da computação podemos ter uma redução de 5% a 20% nos custos para o transporte [GPaolo Toth and Daniele Vigo 2002]. A metodologia utilizada foi uma heurística multi-start com a meta-heurística *Iterated Local Search* (ILS) [Helena R. Lourenço and Olivier C. Martin and Thomas Stützle 2010]. Para a exploração do espaço de soluções e fugir de ótimos locais da busca local foi utilizado o variable neighborhood Search. Todas estruturas de vizinhanças utilizadas foram baseadas por explorar todas características do PRVEF.

2 O PROBLEMA DO ROTEAMENTO DE VEÍCULOS COM ENTREGAS FRACIONÁRIAS

Proposto 1989 [GPaolo Toth and Pierre Trudeau 2008] o problema é representado por um grafo $G = (V, E)$ onde $V = \{0, ..., n\}$ é o conjunto de vértices e o conjunto de arestas é $E = \{\{i, j\} : i, j \in V, i < j\}$. O problema está representado pela Figura 1. A origem é representada pelo vértice 0, uma frota com K veículos, podendo ser limitados (PRVEF-FL) ou ilimitada (PRVEF-FI) com capacidade Q. Os clientes são representados pelos vértices $V^1 = \{1, ..., n\}$ onde a demanda d_i é positivo. O custo c_{ij} está associado aos vértices de E.

O problema tem como objetivo minimizar a soma dos custos desde que: a capacidade do veículo não seja excedida; as demandas

*Nota: Esse artigo está sendo produzido pelo bolsista que teve início a três meses no projeto (Setembro/2021) acumulando assim os resultados preliminares do estudo bibliográfico do tema realizado até o momento. O trabalho está sendo executado ao longo do tempo percorrido da bolsa.

Authors' addresses: Alexandre Dos Santos Mello, alexandre_mello@id.uff.br, Universidade Federal Fluminense, Niterói, Rio de Janeiro, BR; Luiz Satoru Ochi, Universidade Federal Fluminense, Niterói, Rio de Janeiro, BR, satoru@ic.uff.br; Matheus Nohra Haddad, Universidade Federal de Viçosa, Rio Paranaíba, Minas Gerais, BR, matheus.haddad@ufv.br; Igor Machado Coelho, Universidade Federal Fluminense, Niterói, Rio de Janeiro, BR, @ic.uff.br.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2018 Association for Computing Machinery.
0730-0301/2018/8-ART111 $15.00
https://doi.org/10.1145/1122445.1122456

de todos os clientes sejam atendidas; toda rota tenha início e fim na origem.

O número de veículos pode ser calculado pela função (Adicionar a equação no overleaf) soma das demandas de todos clientes divido pela capacidade dos veículos.

$$kmin = \sum_{i=0}^{n} di/Q$$

Fig. 1. Exemplo de uma solução.

Para fazer a validação de uma solução a função objetivo é calculada pela equação [C. Archetti and G. Speranza 2006].

- x_{ij}^k uma variável binária tal que 1 representa a aresta (i,j) visitada pelo veículo k e 0 caso não.
- $y_{ij} \geq 0$ demanda do cliente i entregue pelo veículo k.

$$F_{Arch} = \text{Minimizar} \sum_{k=1}^{K} \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} x_{ij}^k$$

Fig. 2. Função objetico.

2.1 Propriedades do PRVEF

Nas instâncias onde a demanda é 10% maior que a capacidade total dos veículos e utilizando as entregas fracionárias o custo para atender a demanda de todos os clientes é menor, [GPaolo Toth and Pierre Trudeau 2008] and [Moshe Dror, Pierre Trudeau 1990]. uma vez que os clientes podem ter demandas superiores a capacidade do veículo, uma vez que é a entrega pode ser fracionada.

Analisando as soluções ótimas existentes na literatura [Helena R. Lourenço and Olivier C. Martin and Thomas Stützle 2010] and [Moshe Dror, Pierre Trudeau 1990] mostra-se duas propriedades onde o custo $c_{ij} > 0 \forall i, j \in E$

Propriedade 1: Seja os custos c_{ij} atendam a desigualdade triangular $c_{ij} <= c_{ik} + c_{kj}$, logo em uma solução ótima, não terão mais do que um cliente em comum [GPaolo Toth and Pierre Trudeau 2008] and [Moshe Dror, Pierre Trudeau 1990].

Seja um conjunto C de 1 a k clientes, C= $\{c_1, c_2, ..., c_k\}$ onde k sejam as rotas $r_1, r_2, ...,r_k$ >= 2. Temos que $r_1 = c_1, c_2$; $r_2 = c_2, c_3$; $r_{k-1} = c_{k-1}, c_k$; $r_k = c_k, c_1$ chamamos de um ciclo *k-split*, sempre começando e finalizando na origem.

Propriedade 2: Seja os custos c_{ij} atendam a desigualdade triangular, logo em uma solução ótima não conterá *k-split* [GPaolo Toth and Pierre Trudeau 2008] and [Moshe Dror, Pierre Trudeau 1990]..

3 TRABALHOS RELACIONADOS

Wang et al. [Yong Wang and Qin L, Xiangyang Guan, Jianxin Fan, Maozeng Xu, Haizhong Wang 2021] com base no problema clássico, com múltiplos depósitos e janelas de tempo, formularam uma proposta de estabelecer um modelo de otimização de rede colaborativa visando coordenar os serviços de coleta e entrega a partir de N depósitos para determinar as rotas com uma redução no custo computacional. Juntamente com um algoritmo de *Clustering* com o algoritmo de *split* para atribuir o cliente ao seu provedor. Para otimizar as rotas geradas houve a utilização de um Algoritmo Genético com busca tabu.

Melo e Correia [Rafael de Melo Böeger and Omir Correia Alves Junior 2002] também a partir do SDVRPTW (*Split Delivery Vehicle Routing Problem with Time Windows*) propuseram uma abordagem para a resolução do problema, onde transforma a instância recebida pelo SDVRPTW para uma VRPTW sem o método do *split*. Após a aplicação de um Algoritmo Genético de Chaves Aleatórias Viciadas (BRKGA) para a solução do problema, a solução é transformada novamente para uma solução de um SDVRPTW.

De acordo com David e Suzuki [Juan David Cortes, Yoshinori Suzuki 2020], ao utilizar o VRPSD (*Vehicle Routing Problem with Shopment Consolidation*) que consiste em um modelo que é capaz de realizar trocas de remessas entre diferentes veículos em uma rota, podemos ter uma economia de até 10% no custo final além de melhorar as rotas nas ultimas entregas.

4 METODOLOGIA

O modelo consiste na junção do algoritmo Multi Start juntamente com a meta-heurística Iterated Local Search (ILS) [Helena R. Lourenço and Olivier C. Martin and Thomas Stützle 2010] incorporado com o variable neighborhood descent [Pierre Hansen and Nenad Mladenovic 2008].

As heurísticas construtivas, são métodos para formulação de soluções de elemento por elemento. Seus métodos dependem do tipo do problema a ser solucionado, mas a característica gulosa de inserção elemento a elemento respeitando a regra para cada construção da solução baseada no contexto do problema sempre será respeitada

Dentro das heurísticas construtivas, existem as construções gulosas e aleatórias. Dentre elas, está a heurística do vizinho mais próximo, sua construção gulosa, insere elemento a elemento o vizinho mais próximo referente ao problema do Caixeiro Viajante.

A partir de métodos construtivos, as meta-heurísticas tem como objetivo buscar uma solução ótima para cada problema apresentado, sendo implementada de maneira geral diferentemente para cada aplicação.

Como métodos gerais, as meta-heurísticas buscam fugir de ótimos locais através de seu caráter geral utilizando técnicas únicas. Tais técnicas são o maior fator para diferenciarem umas das outras. Divididas em duas categorias, sendo elas as buscas locais e as buscas populacionais, definidas através do método usado para exploração das soluções [Marcone Jamilson Freitas Souza 1997].

As meta-heurísticas voltadas para busca populacional, combinam soluções presentes no conjunto do universo de soluções, a fim de gerar uma melhoria nos resultados da combinação das soluções. Métodos como os Algoritmos Genéticos, Algoritmos Meméticos e o Algoritmo de Colônia de Formigas são exemplos renomados na categoria[Marcone Jamilson Freitas Souza 1997].

As meta-heurísticas baseadas em busca local, procuram gerar uma solução nova para o conjunto utilizando técnicas de alteração, como movimentos na solução corrente pelas estruturas de vizinhança. Cada solução gerada dessa forma é acrescentada na vizinhança da exploração de novas soluções. Técnicas que utilizam desse método como Busca Tabu, *Simulated Annealing*, Busca em Vizinhança Variável (*Variable Neighborhood Search*) e *Iterated Local Search* são os mais frequentes [Marcone Jamilson Freitas Souza 1997].

A meta-heurística *Iterated Local Search* consiste em método para explorar o espaço de soluções por meio de perturbações em ótimos locais. Para que a busca local explore diferentes soluções e seja eficiente, é necessário que a perturbação seja forte, alterando bem a solução e fraca o suficiente para não fugir da região do espaço de soluções geradas [Marcone Jamilson Freitas Souza 1997].

```
procedimento ILS
1  s0 ← GeraSolucaoInicial();
2  s ← BuscaLocal(s0);
3  enquanto (os critérios de parada não estiverem satisfeitos) faça
4      s' ← Perturbacao(histórico, s);
5      s'' ← BuscaLocal(s');
6      s ← CriterioAceitacao(s, s'', histórico);
8  fim-enquanto;
fim ILS;
```

Fig. 3. Algoritmo ILS [Marcone Jamilson Freitas Souza 1997]

Na linha 1 é produzida uma solução inicial, na linha seguinte existe a busca local, responsável por retorna uma solução melhorada. Nas linhas 3 a 6, o laço de perturbação, modificando a solução corrente do laço e o critério de aceitação verifica qual será a próxima solução a ser perturbada. Os critérios de parada do laço podem ser uma melhoria na solução ou um número máximo de perturbações.

4.1 Representação e Avaliação de Soluções

A solução foi representada por duas tabelas: *R* (k linhas e um número variável de colunas) cada linha *r* representa uma rota associada a um veículo e as colunas, a permutação dos clientes. Já a tabela *Z* (k linhas e n colunas) cada linha *r* representa uma linha da tabela *R* onde cada coluna representa os clientes, mostrando assim a porcentagem da demanda de cada cliente. A representação pode ser vista na Figura 1. Imagem da representação editada. Colocar a Figura da representação editada.

4.2 Procedimentos Auxiliares

O primeiro procedimento, *SplitReinsertion* focado em *k-split* [29] está representado pela Figura 4. De início o algoritmo recebe 4 dados de entrada: Uma solução; Um cliente; a demanda $d_i^{'} \leq d_i$; Uma rota proibida $\bar{r}$.

```
Algoritmo 2: SplitReinsertion
1  Procedimento SplitReinsertion(s, i, d''_i, r̄)
2  L ← A ← U ← ∅;
3  Seja K_s o número de veículos em s;
4  sumResidual ← 0;
5  para r ← 1, ..., K_s faça
6      Seja a_r a capacidade residual da rota r;
7      se r ≠ r̄ e a_r > 0 então
8          L ← L ∪ {r};
9          A ← A ∪ {a_r};
10         sumResidual ← sumResidual + a_r;
11     fim se
12 fim para
13 se sumResidual ≥ d''_i então
14     para r ← 1, ..., |L| faça
15         Seja u_r o menor custo adicional para inserir o cliente i em r;
16         U ← U ∪ {u_r};
17     fim para
18     s ← knapGreedyHeuristic(s, i, d''_i, L, A, U);
19 fim se
20 retorne s;
21 fim
```

Fig. 4. Pseudocódigo do SplitReinsertion [Marcos de Melo da Silva, Luiz Satoru Ochi 2013]

Nas Linhas 5 a 12 da Figura 4, se uma rota é diferente de uma rota proibida e sua capacidade residual a_r for maior que 0 a rota é adicionada na lista. Se a condição da linha 13 onde a somas das capacidades residuais não forem suficientes par atendendo o cliente *i* o algoritmo é encerrado. Caso forem suficientes para cada rota pertencente a Lista calcula e insere o custo para inserir o cliente *i* na rota *r* (Linhas 14 a 17). Para determinar qual a inserção do cliente na rota, foi usado o Problema da Mochila na linha 18.

O segundo procedimento *EmptyRoutes* (Figura 5) tem a função de diminuir as rotas de uma solução utilizando o *SplitReinsertion*.

```
Algoritmo 3: EmptyRoutes
1  Procedimento EmptyRoutes(s)
2  enquanto K > K_min faça
3      r̄ ←rota com menor carga em s;
4      enquanto r̄ ≠ ∅ faça
5          i ← primeiro cliente de r̄;
6          s ← SplitReinsertion(s, i, d''_i, r̄);
7          r̄ ← r̄ − {i};
8      fim enquanto
9  fim enquanto
10 retorne s;
11 fim
```

Fig. 5. Pseudocódigo do EmptyRouts [Marcos de Melo da Silva, Luiz Satoru Ochi 2013]

O Terceiro procedimento *ReinsertSingleCustomer* (Figura 6) é responsável por remover rotas unitárias, uma vez que o cliente removido de uma rota, é reinserido em uma segunda rota podendo gerar uma solução com um menor custo.

Algoritmo 4: ReinsertSingleCustomer

```
1  Procedimento ReinsertSingleCustomer(s)
2  aplicarSplit ← falso;
3  f'' ← f(s);
4  para r ← 1, ..., |s| faça
5      se |r| = 1 então
6          i ← único cliente em r;
7          s' ← SplitReinsertion(s, i, d_i, r);
8          se f(s') < f'' então
9              s'' ← s';
10             f'' ← f(s');
11             aplicarSplit ← verdadeiro;
12         fim se
13     fim se
14 fim para
15 se aplicarSplit = verdadeiro então
16     s ← s'';
17     Vá para a linha 2;
18 fim se
19 retorne s;
20 fim
```

Fig. 6. Pseudocódigo do ReinsertSingleCustomer [Marcos de Melo da Silva, Luiz Satoru Ochi 2013]

Nas linhas 4 a 14 para as rotas da solução que que são unitárias é aplicado o método, quando o custo da nova solução é menor (linhas 8 a 12) solução corrente é salva. Caso não seja mais possível encontrar novas solução viáveis o algoritmo encerra.

4.3 Métodos Construtivos

Para a construções das soluções iniciais são usadas as heurísticas de inserção sendo incremental cliente por cliente. Para definir quais clientes serão inseridos em quais rotas são utilizados dois critérios, a inserção mais próxima (CIMPV) e a inserção viável mais barata (CIMBVM) e também duas estratégias, a inserção sequencial (EISS) e a inserção paralela (EIP) [10].

4.3.1 Critério de Inserção. O custo ao inserir um cliente em uma rota é calculado pela equação $g(k)$ dividida em duas etapas. A primeira responsável por calcular o custo da inserção do cliente i para os seus adjacentes j E k. A segunda etapa é responsável por penalizar aquelas soluções onde os clientes para serem inseridos estão muitos distantes da origem.

$g(k) = (c^r_{ik} + c^r_{kj} - c^r_{ij}) - \gamma(c^r_{0k} + c^r_{k0})$

O custo da inserção é dado pela equação: $g(k) = c^r_{ik}$; onde o cliente k é inserido depois do cliente i. Nos dois critérios, a inserção do cliente está associada com o menor custo,

$i_{min} < -argmin\{g(k)|k \in CL\}$ [10].

4.3.2 Estratégias de Inserção. Para estratégias de inserção são consideradas duas estratégias, a inserção sequencial (EISS) e inserção paralela (EIP). A principal diferença entre elas é o número de rotas que receberão uma inserção. No critério CIMPV o valor de γ é selecionado aleatoriamente no intervalo de 0.1 a 1.0[10].

A estratégia EISS representado pela Figura 7 utiliza de uma rota. Nas linhas 2 a 4 do algoritmo é feita a definição do γ. O laço da linha 6 a 22 acontece enquanto existem clientes para serem inseridos. Uma vez que não há possibilidades de incremento de clientes na rota, um veículo extra é inserido pelas linhas 17 a 21 recomeçando o laço, dessa forma todos os clientes serão atendidos.

Algoritmo 6: InserçãoSemiSequencial

```
1  Procedimento InserçãoSemiSequencial(s, LC, CritérioInserção, gammaInterval)
2  se CritérioInserção = CIMBVM então
3      γ ← valor ∈ gammaInterval, selecionado aleatoriamente;
4  fim se
5  r_0 ← 1;
6  enquanto LC ≠ ∅ faça
7      clienteInserido ← falso;
8      para r ← r_0, ..., |s| faça
9          se LC ≠ ∅ e ∃i ∈ LC que pode ser inserido na rota r então
10             Calcule o custo g(i), ∀i ∈ LC, conforme CritérioInserção;
11             i' ← arg min_{i ∈ LC}{g(i)};
12             s[r] ← s[r] ∪ {i'};
13             LC ← LC − {i'};
14             clienteInserido ← verdadeiro;
15         fim se
16     fim para
17     se |LC| > 0 e clienteInserido = falso então
18         Seja r' uma rota vazia;
19         s ← s ∪ {r'};
20         r_0 ← r';
21     fim se
22 fim enquanto
23 retorne s;
24 fim
```

Fig. 7. Pseudocódigo do InserçãoSemiSequencial [Marcos de Melo da Silva, Luiz Satoru Ochi 2013]

A estratégia EIP representado pela Figura 8, tendo o γ também definido pelas linhas 2 a 4, seu laço executará enquanto houver clientes na LC para serem inseridos, porém na inserção, o algoritmo leva em conta todas as possíveis rotas ao inserir um cliente. Similar o EISS, quando não for mais possível adicionar mais clientes a rota, será inserido um novo veículo (linhas 17 a 21) atendendo todas as demandas e clientes.

Algoritmo 7: InserçãoParalela

```
1  Procedimento InserçãoParalela(s, LC, CritérioInserção, gammaInterval)
2  se CritérioInserção = CIMBVM então
3      γ ← valor ∈ gammaInterval, selecionado aleatoriamente;
4  fim se
5  r_0 ← 1;
6  enquanto LC ≠ ∅ faça
7      clienteInserido ← falso;
8      se LC ≠ ∅ e ∃i ∈ LC que pode ser inserido em s então
9          Calcule o custo g(i, r), ∀i ∈ LC e ∀r ∈ {r_0, ..., |s|}, conforme CritérioInserção;
10         (i', r') ← arg min_{i ∈ LC, r ∈ {r_0,...,|s|}}{g(i, r)};
11         s[r'] ← s[r'] ∪ {i'};
12         LC ← LC − {i'};
13         clienteInserido ← verdadeiro;
14     fim se
15     se |LC| > 0 e clienteInserido = falso então
16         Seja r'' uma rota vazia;
17         s ← s ∪ {r''};
18         r_0 ← r'';
19     fim se
20 fim enquanto
21 retorne s;
22 fim
```

Fig. 8. Pseudocódigo do InserçãoParalela

4.4 Busca Local

O VND mais usado visa estruturas de vizinhanças pré-estabelecidas em uma sequência, porém estudos recentes comprovam que uma abordagem aleatória na exploração das estruturas de vizinhança vem se mostrando eficiente, diversificando o resultado das buscas. Souza et al. (2010) e Subramanian et al. (2010) definiram esse método

de Random Variable Neighborhood Descent (RVND) A Busca local será realizada através desse método, a fim de encontrar melhores soluções através da busca local.

Algoritmo 8: RVND

1 **Procedimento** RVND(s)
2 Inicialize a lista de vizinhanças NL;
3 Atualize as estruturas de dados auxiliares;
4 **enquanto** $NL \neq \emptyset$ **faça**
5 Escolha uma vizinhança $N^{(h)} \in NL$ aleatoriamente;
6 Encontre o melhor vizinho s' de $s \in N^{(h)}$;
7 **se** $f(s') < f(s)$ **então**
8 $s \leftarrow s'$;
9 $s \leftarrow$ CheckAndRepair(s);
10 $s \leftarrow$ IntraRouteRVND(s);
11 Atualize NL;
12 **senão**
13 Remova $N^{(h)}$ de NL;
14 **fim se**
15 Atualize as estruturas de dados auxiliares;
16 **fim enquanto**
17 **retorne** s;
18 **fim**

Fig. 9. Pseudocódigo do RVND

De acordo com o Pseudocódigo acima o RVND inicia com a lista (NL) de estruturas de vizinhança, a cada laço das linhas 4 a 16, é selecionado uma estrutura aleatória da lista e é gerado um vizinho através do movimento. Caso o novo vizinho tenha um menor custo, o vizinho passa ser a nova solução e o laço é reiniciado. O critério de parada se dá quando a lista de estruturas for igual a zero.

4.4.1 Vizinhanças Inter Rotas. Para as vizinhanças inter rotas são utilizadas 10 estruturas. Cinco das 10 iniciais são do tipo λ-*interchanges*, que fazer a troca de λ clientes entre duas rotas diferentes. A estrutura seis (*Cross-exchange*) irá realizar a troca de um segmento da rota. Já as estruturas sete e oito sendo *Swap(1,1)* e *Swap(2,1)* referentes a fração. A Nona estrutura (*Route Addition*) sendo específica do problema

A primeira estrutura *Shift (1,0)*–$N^{(1)}$– Remoção de um cliente de uma rota r_1 e inserindo em uma rota r_2. *Swap (1,1)*–$N^{(2)}$– é realizada a troca entre o cliente i da rota r_1 com o cliente j da r_2. *Shift (2,0)*–$N^{(3)}$– Remoção de dois clientes adjacentes de uma rota r_1 e inserindo em uma rota r_2. *Swap (2,1)*–$N^{(4)}$– é realizada a troca entre dois clientes adjacentes i e j da rota r_1 com um cliente k da r_2. *Swap (2,2)*–$N^{(5)}$– é realizada a troca entre dois clientes adjacentes i e j da rota r_1 com dois clientes adjacentes k e l da r_2. Estrutura *Cross* - $N^{(6)}$– O arco entre os clientes i e j da r_1 e o arco entre os clientes k e l da r_2 também são removidos. Logos após isso é criado um novo arco entre os clientes i e l e entre os clientes k e j.

*Swap (1,1)**–$N^{(7)}$– uma extensão do *Swap (1,1)* ao modificar a quantidade de demanda de te i e j são considerados dois casos onde: $\forall i \in r_1$ e $\forall j \in r_2$; (i) $d_i > d_j$ o cliente da rota r_2 é inserido em r_1 e um copia do cliente i é substitui o cliente j na r_2. (ii) $d_j > d_i$ o cliente i da rota r_1 é inserido na r_2 e o cliente i é substituído por uma cópia do cliente j. Como podemos ver na Imagem a seguir, onde a demanda do cliente 8 é maior que a demanda do cliente 5.

*Swap (2,1)**–$N^{(8)}$– uma extensão do *Swap (2,1)*. Dado dois clientes (i, j): $\forall i \in r_1$ e $\forall j \in r_1$ e $\forall k \in r_2$. (i) $(d_i+d_j) > k_k$. K será inserido na rota r_1 e então na r_2 no lugar de k é inserido o cliente i e uma cópia

Fig. 10. Estrutura de Vizinhança *Swap (1,1)**–$N^{(7)}$–. Caso (i)

do cliente j com a nova demanda de $d'_j = (d_k - d_i)$; (ii) $k_k > (d_i + d_j)$. i e j serão adicionados na rota r_2 e na r_1 serão substituídos por uma cópia do cliente k com a nova demanda de $d'_k = (d_i - d_j)$;

Essa estrutura está ilustrada na Figura 11 a seguir.

Fig. 11. Estrutura de Vizinhança *Swap (2,1)**–$N^{(8)}$

RouteAddition–$N^{(9)}$– Para todo cliente i pertencente a mais de uma rota, inicialmente é removido das fotas que aparecer, logo após a remoção é realizada uma tentativa de combinar as rotas gerando 3 novas rotas. O cliente anterior de i na r_1 e na r_2 (i-1) conecta-se direto para o cliente $i+1$. A criação dessa rota [29] está demonstrada pelo imagem a seguir

K-Split–$N^{(10)}$– Para todos os clientes i presentes nas soluções ele é removido, para encontra a melhor posição de i em todas as rotas, mesmo que seja parte de sua demanda. É utilizado o problema da mochila, para verificar e selecionar as melhoras rotas para o cliente i.

4.4.2 Vizinhanças Intra Rotas. Além das estruturas de vizinhanças inter Rotas, existem as estruturas clássicas intra Rotas (N') que são abordadas tanto para o Problema do Caixeiro Viajante (PCV) como o Problema de Roteamento de Veículos, já bem exploradas pela literatura.

- *Exchange*–$N'^{(1)}$– Troca entre dois clientes
- *2-opt*–$N'^{(2)}$– Remoção e adição de dois novos arcos não adjacentes
- *Reinsercão*–$N'^{(3)}$– Um cliente i é removido de uma posição e adicionado em outra posição da mesma rota.
- *Or-opt2*–$N'^{(4)}$– Dois clientes são removidos de suas posições e adicionas em outra posição da rota
- *Or-opt3*–$N'^{(5}$– Três clientes são removidos de suas posições e adicionas em outra posição da rota

4.5 Mecanismos de Pertubação

Todas as pertubações na solução são geradas a partir da utilização das estruturas de vizinhança durante a busca local.

- *Multiple-k-Split* - $P^{(1)}$ - A partir de uma lista com 5 a 7 clientes é gerada aleatoriamente e o *k-split* é aplicado para cada cliente.
- *Route Addition + k - split* - $P^{(2)}$ - Seja um cliente que está sendo visitado por mais de uma rota, será realizado um movimento aleatório da estrutura *Route Addition* que irá dividir dando incio ao *Multiple - - Split.*
- *Multiple-Swap(1,1)** - $P^{(3)}$ - A estrutura de vizinhança $N^{(7)}$ é executada aleatoriamente várias vezes.

4.6 Algoritmo

Presentando pelo Pseudocódigo a baixo, o SplitILS é combinação de uma heurística *multi-start* baseada em um ILS com buscas locais (RVND). A solução é inicialmente construída pela linha 4. Durante as linhas 7 a 18 o laço é escutado enquanto as iterações não melhoram utilizando o RVND. Assim que a busca local é encerra, com a lista de estrutura vazia, a solução encontrada é um ótimo local caso seja frota limitada. Caso o critério de aceitação seja verdadeiro o laço da linha 12 a 15 é reiniciado. Agora a solução será perturbada ela linha 16. Ao final da execução, o algoritmo retorna a melhor solução encontrada.

Algoritmo 9: SplitILS

```
1  Procedimento SplitILS(I_Max, I_ILS)
2  f* ← ∞;
3  para i ← 1, ..., I_Max faça
4      s ← GSolInicial();
5      s' ← s;
6      iterILS ← 0;
7      enquanto iterILS < I_ILS faça
8          s ← RVND(s);
9          se PRVEF-FL então
10             s ← EmptyRoutes(s);
11         fim se
12         se f(s) < f(s') então
13             s' ← s;
14             iterILS ← 0;
15         fim se
16         s ← Perturba(s');
17         iterILS ← iterILS + 1;
18     fim enquanto
19     se f(s') < f* então
20         s* ← s';
21         f* ← f(s');
22     fim se
23 fim para
24 retorne s*;
25 fim
```

Fig. 12. Pseudocódigo do SplitILS

5 RESULTADOS DA LITERATURA

Os resultados obtidos estão mostrados pela Figura a seguir

- Instâncias representa o número de problemas-testes
- Melhores representa onde o algoritmo SplitILS encontrou melhores soluções
- Empates onde houve resultados iguais aos da Literatura
- *Gap* diferença de melhora. O **Gap** negativo indica onde o custo encontrado foi menor

Conjunto	#Instâncias	#Melhores	#Empates	Avg. Gap (%)	Avg. Time (s)	Avg. Time BKS (s)
Belenguer [14][a,c]	25	18	7	−1,87	63,77	1,50
Belenguer [14][a]	25	22	3	−1,46	75,97	1,69
Campos [19][a]	49	44	5	−2,23	925,59	3,06
Archetti [13][a]	42	36	-	−2,81	1159,19	2,68
Chen [21][a]	21	7	5	−0,02	504,44	16,41*
Belenguer [14][b,c]	25	13	12	−0,32	56,87	9,47
Belenguer [14][b]	25	22	3	−1,13	74,51	1,39
Campos [19][b]	49	40	9	−0,41	872,02	63,55
Archetti [13][b]	42	36	5	−1,22	1062,86	44,74*
Chen [21][b]	21	5	6	−0,03	591,62	14,67*
Total	324	243	55			

a - PRVEF-FL
b - PRVEF-FI
c - Custos arredondados
* - Desconsiderando as instâncias em que o SplitILS não foi capaz de encontrar ou melhorar a BKS

Fig. 13. Resultados Gerais do SplitILS

A abordagem apresentada possui uma boa eficácia, é capaz de melhoras os resultados já presentes na literatura em 65 soluções e de empatar em 18

6 CONCLUSÃO E TRABALHOS FUTUROS

Como trabalhos futuros e dando continuidade ao projeto de iniciação, vemos oportunidade de desenvolver duas vertentes, a primeira seria a proposta de um algoritmo para solucionar o problema de roteamento de veículos de rota única com entregas obrigatórias fracionárias e coletas seletivas. Levando em conta que a coleta não seria obrigatória e todas as vezes que ela acontecer por um veículo será descontado no custo total da solução como uma bonificação. A segunda vertente similar à primeira porém sendo parcialmente obrigatória a coleta e existe uma cota mínima de coletas que cada veículo é obrigado a fazer.

REFERENCES

C. Archetti and G. Speranza. 2006. *A Tabu Search Algorithm for the Split Delivery Vehicle Routing Problem.*

G. B. Dantzig and J. H. Ramser. 208. *The Truck Dispatching Problem.*

GPaolo Toth and Daniele Vigo. 2002. *The Vehicle Routing Problem.*

GPaolo Toth and Pierre Trudeau. 2008. *Savings by Split Delivery Routing.*

Helena R. Lourenço and Olivier C. Martin and Thomas Stützle. 2010. *Iterated Local Search: Framework and Applications.*

Juan David Cortes, Yoshinori Suzuki. 2020. *Vehicle Routing with Shipment Consolidation.*

Marcone Jamilson Freitas Souza. 1997. *Inteligência computacional para otimizaçao.*

Marcos de Melo da Silva, Luiz Satoru Ochi. 2013. *Uma Heurística Baseada em Irerated Local Search Para o Problema do Roteamento de Ve ´culos com Entregas Fracionárias.*

Moshe Dror, Pierre Trudeau. 1990. *Split delivery routing.*

Olli Bräysy and Michel Gendreau. 2005. *Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms.*

Pierre Hansen and Nenad Mladenovic. 2008. *Variable neighbourhood search: methods and applications.*

Rafael de Melo Böeger and Omir Correia Alves Junior. 2002. *RESOLUÇÃO DO PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM ENTREGAS FRACIONADAS E JANELAS DE TEMPO POR MEIO DE UM ALGORITMO GENÉTICO COM CHAVES ALEATÓRIAS VICIADAS.*

Yong Wang and Qin L, Xiangyang Guan, Jianxin Fan, Maozeng Xu, Haizhong Wang. 2021. *Collaborative multi-depot pickup and delivery vehicle routing problem with split loads and time windows.*

Resolvendo instâncias de entregas urbanas reais com janelas de tempo baseadas no benchmark loggiBUD e aplicando um Branch-Cut-and-Price via VRPsolver

Juan Rosas
Prog. Pós-Graduação em Informática
Instituto de Computação - UFAM
Manaus, AM, Brasil
juan.rosas@icomp.ufam.edu.br

Thailsson Clementino
Bac. Ciência da Computação
Instituto de Computação - UFAM
Manaus, AM, Brasil
thailsson.clementino@icomp.ufam.edu.br

Rosiane de Freitas
Instituto de Computação
Univ. Federal do Amazonas - UFAM
Manaus, AM, Brasil
rosiane@icomp.ufam.edu.br

ABSTRACT

Neste artigo, o problema de roteamento de veículos com janelas de tempo (do inglês, *Vehicle Routing Problem with Time Windows* - VRPTW) é abordado, como forma de incorporar características de dinamicidade da cadeia logística de entregas expressas, explorando aspectos teóricos que melhor definem o problema e suas instâncias, e propiciando ferramental para estratégias que contribuam na resolução de instâncias de entregas urbanas reais. O VRPTW consiste em determinar rotas de custo mínimo que devem ser executadas por uma frota de veículos homogêneos respeitando suas capacidades e as janelas de tempo de entregas de cada cliente. Uma estratégia que utiliza o método de Branch-Cut-and-Price através da ferramenta VRPsolver foi aplicada em dois conjuntos de instâncias, sendo a primeira as instâncias clássicas artificiais de Solomon e, a segunda, instâncias reais brasileiras do benchmark loggiBUD, que foram adaptadas para conter janelas de tempo. Os resultados reforçam a adequabilidade da proposta, que se mostrou de melhor desempenho quando as janelas de tempo foram de intervalos mais justos.

CCS CONCEPTS

• **Mathematics of computing → Combinatorial optimization**; **Solvers**; • **Applied computing → Operations research**.

KEYWORDS

branch-cut-and-price method, graph theory, integer programming, VRPsolver, Vehicle Routing Problem with Time Windows

ACM Reference Format:
Juan Rosas, Thailsson Clementino, and Rosiane de Freitas . 2021. Resolvendo instâncias de entregas urbanas reais com janelas de tempo baseadas no benchmark loggiBUD e aplicando um Branch-Cut-and-Price via VRPsolver. In *Workshop PBP-Loggi: Resultados do Programa de Bolsas de Pesquisa (PBP-LOGGI)., Dez 08, 2021.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Workshop PBP-Loggi, Dez, 2021,
© 2021 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUÇÃO

Os canais de vendas virtuais têm se tornado cada vez mais relevantes para clientes e empresas. Nesse sentido, a pandemia da COVID-19 e o consequente isolamento social imposto desde o início de 2020, levou à mudanças significativas nos hábitos de consumo, em particular, à migração massiva para as compras online [1], o que gerou desafios significativos para a cadeia logística. Nesse novo cenário, devido a competitividade do mercado, as empresas ampliaram a oferta e demanda de entregas no mesmo dia com baixo custo [4].

Um desafio logístico em especial consiste em alocar os pedidos a serem entregues, em veículos com determinadas capacidades, de modo a realizar o percurso no menor tempo ou distância possível, saindo e retornando ao depósito após passar por todos os pontos de entrega, sendo conhecido na literatura como o clássico Problema de Roteamento de Veículos (do inglês, *Vehicle Routing Problem* - VRP). O VRP e suas variantes têm sido objeto de intensa investigação na pesquisa operacional e afins [3].

Em aplicações reais do VRP, várias restrições operacionais e de dinamicidade adicionais precisam ser levadas em consideração, como o período do tempo em que cada cliente poderá receber seu pedido. Uma aplicação real do problema é encontrado na empresa de logística de entregas expressas Loggi, que se baseia em um modelo de economia compartilhada, onde os motoristas independentes usam a plataforma online da Loggi para aceitar e fazer entregas com seus próprios veículos [4]. O maior desafio da Loggi é criar uma rede de logística sustentável para entregas no mesmo dia, desafio no qual é possível observar o problema real do roteamento de veículos com janela de tempo e sua complexidade de permissibilidade dos prazos de entrega, ou janelas de tempo, decorrentes do fato de que alguns clientes impõem prazos de entregas e/ou restrições de tempo de entrega mais cedo ou mais tarde [12]. Analisando as necessidades deste cenário, uma questão de pesquisa pode ser formulada: será que a aplicação combinada de teoria de escalonamento com modelos clássicos de problemas de roteamento de veículos capacitados podem prover modelos realísticos mais robustos teoricamente e possibilitar a geração de estratégias exato-heurísticas mais eficientes? Com base nesta hipótese de pesquisa, que embasa a pesquisa geral envolvida no Programa de Bolsas de Pesquisa (PBP) da LOggi com a UFAM, em desenvolvimento pelo grupo CNPq de Otimização, Algoritmos e Complexidade Computacional (ALGOX) do IComp/UFAM. Neste trabalho são apresentados os primeiros resultados, com a incorporação de janelas de tempo ao problema e instâncias reais brasileiras do benchmark loggiBUD e aplicação do método de Branch-Cut-ANd-Price (BCP) via ferramenta VRPsolver.

O restante deste trabalho está organizado como segue. Na Seção 2 é formalmente descrito o VRPTW. Na Seção 3 são apresentados os trabalhos relacionados. Na Seção 4, a formulação matemática em programação inteira e o método BCP que a manipula são descritos, para utilização na ferramenta VRPsolver. Na Seção 5, os resultados obtidos são apresentados e discutidos. Por fim, na Seção 6 são tecidas as considerações finais sobre o trabalho.

2 PROBLEMA DO ROTEAMENTO DE VEÍCULOS COM JANELAS DE TEMPO - VRPTW

O problema de roteamento de veículos com janelas de tempo (*Vehicle Routing Problem with Time Windows* - VRPTW) é uma variação do VRP clássico, onde cada cliente possui um intervalo de tempo para ser atendido e cada veículo possui uma capacidade de carga máxima que pode transportar. Esse intervalo de tempo, chamado de janela de tempo é composto pela hora de início e fim de atendimento disponível para cada cliente, ou seja, a entrega não pode ser antes do momento inicial, nem depois do momento final. A restrição de capacidade veicular exige que as entregas feitas em cada rota não excedam a capacidade máxima de cada veículo.

O objetivo do VRPTW consiste em encontrar rotas de tempo total mínimo que atendam todos os clientes, respeitando as restrições de capacidade dos veículos e as janelas de tempo utilizando o mínimo de veículos para a realização do roteamento.

Seja N o número de clientes a serem atendidos, onde para cada cliente uma entrega $i \in \{1, 2, ..., N\}$ deve ser realizada. A cada cliente está associada uma janela de tempo $[a_i, b_i], a_i < b_i$, que define o horário mais cedo e o horário mais tarde em que o serviço pode ser executado, uma quantidade $q_i > 0$ de carga a ser entregue e um tempo de serviço $s_i > 0$.

No depósito de saída existem NV veículos disponíveis, cada um com uma capacidade k. O depósito possui demanda zero, tempo de serviço zero e sua janela de tempo se estende desde a hora zero até uma hora grande o suficiente para que um veículo retorne de qualquer cliente que esteja atendendo. Os vértices 0 e $N + 1$ representam, respectivamente, a base de saída (depósito) e a de chegada dos veículos (depósito).

O VRPTW pode ser descrito por um grafo $G(V, A)$, onde V é o conjunto de vértices representando todos os clientes e o depósito, e A é o conjunto de arestas que representam os caminhos que ligam os nós. O grafo G é modelado como um grafo completo, ou seja, todos os vértices tem arestas que os conectam com todos os outros. Cada aresta é representada por um arco (i, j) que possui um custo c_{ij} associado indicando o tempo necessário para se percorrer cada aresta, calculado como o comprimento de cada arco.

Um dos modelos teóricos mais conhecidos para o VRPTW é a formulação em programação inteira proposta por Kohl et al.(1999) [5]. Esse modelo visa a minimização do tempo total de entrega das cargas, considerando as restrições de janelas de tempo e de capacidade dos veículos que são determinísticas a priori. A formulação matemática do VRPTW compreende as seguintes variáveis de decisão:

$$X_{ij}^v = \begin{cases} 1, \text{se } j \text{ é atendido após } i \text{ pelo veiculo } v \\ 0, \text{caso contrário} \end{cases}$$

T_i = horário de atendimento em $i, i \in \{1, 2, ..., N\}$

O modelo matemático do VRPTW é apresentado a seguir:

$$min \sum_{v=1}^{NV} \sum_{i=0}^{N} \sum_{j=1}^{N+1} c_{ij} X_{ij}^v \tag{1a}$$

$$S.t. \sum_{v=1}^{NV} \sum_{j=1}^{N+1} X_{ij}^v = 1, \qquad i = 1, 2, ..., N; i \neq j \tag{1b}$$

$$\sum_{j=0}^{N+1} X_{0j}^v = 1, \qquad v = 1, 2, ..., NV \tag{1c}$$

$$\sum_{i=0}^{N} X_{ij}^v - \sum_{i=1}^{N+1} X_{ji}^v = 0 \qquad j = 1, ..., N; v = 1, ..., NV \tag{1d}$$

$$\sum_{i=0}^{N+1} X_{i,N+1}^v = 0 \qquad v = 1, ..., NV \tag{1e}$$

$$a_i \leq T_i \leq b_i \qquad i = 1, ..., N \tag{1f}$$

$$T_i + s_i + c_i - T_j \leq (1 - X_{ij}^v) M_{ij} \qquad (i, j) \in A; v = 1, ..., NV \tag{1g}$$

$$\sum_{i=0}^{N} \sum_{j=1}^{N} q_j X_{ij}^v \leq K \qquad v = 1, ..., NV \tag{1h}$$

$$X_{ij}^v \in \{0, 1\} \qquad (i, j) \in A; v = 1, ..., NV \tag{1i}$$

A função objetivo (1a) representa o tempo total a ser minimizado. As restrições (1b) asseguram que cada cliente seja visitado uma única vez e por um único veículo; as restrições (1c) asseguram que todo veículo inicie uma rota a partir do depósito; as restrições (1d) impõem que todo veículo deve entrar e sair de um cliente; as restrições (1e) impõem que todo veículo termine uma rota no depósito. Caso o veículo não seja utilizado, ele segue o caminho do arco direto ligando a base de partida (nó 0) à base de chegada (nó $N + 1$). As restrições (1f) impõem que o horário de início de atendimento de cada nó ocorra dentro da sua respectiva janela de tempo. As restrições (1g) estabelecem a relação entre o horário de partida do veículo a partir de um cliente e seu sucessor imediato, garantindo também a eliminação de sub-rotas. As restrições de capacidade de carga são dadas pela expressão (1h). Por fim, as restrições (1i) asseguram a integralidade da solução. As restrições (1g) foram obtidas a partir da linearização das restrições originais não lineares ($X_{ij}^v(T_i + s_i + c_i - T_j) \leq 0,\ i = 1, 2, ..., N; j = 1, 2, ..., N; v = 1, 2, ..., NV$), a qual foi possível através da condição de integralidade pelas restrições (1i). O valor da constante M_{ij} nas restrições pode ser substituído por $max(b_i + s_i + c_{ij} - a_i, 0), i, j \in \{1, 2, ..., N\}$

3 TRABALHOS RELACIONADOS

Existem vários trabalhos que abordam o problema de roteamento de veículos com janelas de tempo, com pequenas variações entre eles

e diferentes abordagens de resolução. Nesta seção um compêndio dos principais resultados é apresentado.

Solomon (1987)[12], formaliza o VRPTW (com janelas de tempo) e propõe uma estrutura de instâncias artificiais que passou a ser referência para esta variação do VRP.

Yassen (2017) [14] apresenta um algoritmo híbrido adaptativo para o VRPTW. O HSA (Harmony Search Algorithm) incorpora um mecanismo de seleção adaptativa para selecionar um algoritmo de busca local adequado a ser aplicado.Segundo os autores, o método é competitivo e foi demonstrado que o mecanismo de seleção pode ajudar efetivamente o HSA a selecionar de forma adaptativa uma busca local adequada, durante o processo de solução do problema.

Wang e Lin (2017) [13], desenvolveram uma metodologia para incorporar características de aplicações reais ao VRPTW, tratando os tempos de viagem como variáveis aleatórias, seguindo algumas funções de distribuição de probabilidade pré-especificadas. O modelo produz consistentemente resultados com menos clientes não atendidos, cargas de trabalho menos desequilibradas e, portanto, menor custos totais.

Nasri (2021) [7], examina uma versão robusta do VRPTW em que os tempos de viagem e de serviço são ambos sujeitos à incerteza. Para tanto aplicam um ALNS *(Adaptive Large Neighborhood Search)*, em conjunto com p método de Monte Carlos e multithread [11].

4 ESTRATÉGIA ELABORADA PARA RESOLUÇÃO DO VRPTW

Atualmente, métodos de Branch-Cut-and-Price são as abordagens dominantes no contexto de roteamento de veículos [2]. Infelizmente, projetar e codificar um algoritmo Branch-Cut-and-Price complexo e sofisticado pode ser uma tarefa altamente exigente com vários meses de trabalho de uma equipe qualificada. Por isso, neste trabalho resolveu-se utilizar um framework proposto para construção de modelos robustos sem que se use meses de trabalho, o VRPSolver [9].

Na Subseção 4.1 é explicado como funciona o VRPSolver e a teoria relacionada a em sua construção. Na subseção 4.2 é explicado a modelagem utilizada para solucionar o VRPTW.

4.1 VRPSolver

A ideia do solver genérico (VRPSolver) é definir um modelo genérico tal que, quando se consegue encaixar um problema de otimização nesse modelo genérico, então ele pode ser resolvido por um algoritmo de Branch-Cut-and-Price, onde todos os subproblemas são modelados como um caminho mínimo com restrição de recurso (do inglês Resource Constrained Shortest Path RCSP [10]). Abaixo é definido resumidamente esse modelo genérico.

4.1.1 Grafos para os Subproblemas RCSP. Seja K o conjunto de subproblemas. É definido grafos direcionados $G^k = (V^k, A^k), k \in K$. $V = \cup_{k\in K} V^k$ e $A = \cup_{k\in K} A^k$. É definido um conjunto de recursos R^k. Para cada $r \in R^k$ e $a \in A^k$, $q_{a,r} \in R$ é o consumo do recurso r no arco a. Além disso é definido um intervalo para o consumo acumulado do recurso $[l_{a,r}, u_{a,r}]$, para cada $r \in R^k$ e $a \in A^k$ (Esse intervalo também pode ser definido nos vértices, nesse caso $[l_{a,r}, u_{a,r}] = [l_{v,r}, u^{v,r}$ para todo arco $a \in \delta^-(v)$).

Um caminho com restrição de recurso $p = (v^k_{source} = v_0, a_1, v_1, ..., a_{n-1}, v_{n-1}, a_n, v_n = v^k_{sink})$ sobre o grafo G^k é factível se respeitar os intervalos para o recurso acumulado ao visitar cada um dos vértices i.

4.1.2 Formulação Mestre. Para todo $a \in A$ e $p \in P$, h^p_a indica quantas vezes o arco a aparece no caminho p. Dado $c \in R^{n_1}$, $f \in R^{n_2}$, $\alpha \in R^{m\times n_1}$, $\beta \in R^{m\times n_2}$, e $d \in R^m$, o problema genérico é formulado como o seguinte:

$$min \sum_{j=1}^{n_1} c_j x_j + \sum_{s=1}^{n_2} f_s y_s \tag{2a}$$

$$S.t. \sum_{j=1}^{n_1} \alpha_{ij} x_j + \sum_{s=1}^{n_2} \beta_{is} y_s \geq d_i, \qquad i = 1, ..., m, \tag{2b}$$

$$x_j = \sum_{k\in K} \sum_{p\in P^k} \left(\sum_{a\in M(x_j)} h^p_a \right) \lambda_p, \qquad j = 1, ..., n_1, \tag{2c}$$

$$L^k \leq \sum_{p\in P^k} \lambda_p \leq U^k, \qquad k \in K, \tag{2d}$$

$$\lambda_p \in Z_+, \qquad p \in P, \tag{2e}$$

$$x_j \in Z, \qquad j = 1, ..., \overline{n}_1, \tag{2f}$$

$$y_s \in Z, \qquad s = 1, ..., \overline{n}_2. \tag{2g}$$

onde x_j, $1 \leq j \leq n_1$, y_s, $1 \leq s \leq n_2$, e $\lambda_p, p \in P$ são variáveis. Os primeiros $\overline{n}_1$ x e os primeiros $\overline{n}_2$ y são inteiros; Todos os λ são inteiros não negativos. A Equação 2a define uma função objetivo linear, onde c e f são vetores de custo. As Inequações 2b definem m restrições lineares sobre as variáveis α e β A relação entre as variáveis x e λ é dado por 2c. Para cada $k \in K$, L^k e U^k são dados como limite inferior e superior de número de caminhos de G^k em uma solução.

Uma solução factível para a Formulação 2 é composto por um conjunto de caminhos, $p \in P$ com multiplicidade λ_p na solução, e decisões adicionais representados por valores atribuidos as variáveis y_s , $s = 1, ..., n_2$. Os valores de x_j, $j = 1, ..., n_1$ são completamente definidos como uma função de caminhos que compõem a solução através dos mapeamentos. Eliminando as variáveis x e relaxando a integridade das restrições:

$$min \sum_{k\in K} \sum_{p\in P^k} \left(\sum_{j=1}^{n_1} c_j \sum_{a\in M(x_j)} h^p_a \right) \lambda_p + \sum_{s=1}^{n_2} f_s y_s \tag{3a}$$

$$S.t. \sum_{k\in K} \sum_{p\in P^k} \left(\sum_{j=1}^{n_1} \alpha_{ij} \sum_{a\in M(x_j)} h^p_a \right) \lambda_p + \sum_{s=1}^{n_2} \beta_{is} y_s \geq d_i, \quad i = 1, ..., m, \tag{3b}$$

$$L^k \leq \sum_{p\in P^k} \lambda_p \leq U^k, \quad k \in K, \tag{3c}$$

$$\lambda_p \geq 0, \quad p \in P. \tag{3d}$$

O problema mestre 3 é resolvido por geração de coluna. Onde π_i é a variável dual relativo a Restrição 3b, v^k_+ e v^k_-, $k \in K$, são as variáveis duais em relação a Restrição 3c. O custo reduzido da aresta $a \in A$ é definido como:

$$\bar{c}_a = \sum_{j \in M-1(a)} c_j - \sum_{i=1}^{m} \sum_{j \in M^{-1}(a)} \alpha_{ij} \pi_i. \tag{4}$$

O custo reduzido de um caminho $p = (v^k_{source} = v_0, a_1, v_1, ..., a_{n-1}, v_{n-1}, a_n, v_n = v^k_{sink}) \in P_k$ é:

$$\bar{c}_p = \sum_{j=1}^{n} \bar{c}_{a_j} - v^k_+ - v^k_-. \tag{5}$$

Assim o subproblema de pricing, corresponde a encontrar, para cada $k \in K$, um caminho $p \in P^k$ com o menor custo reduzido. Para mais informações de implementação é necessário ler o trabalho original do VRPSolver [9].

4.2 Modelo

O modelo construído para resolver o VRPTW utilizando o VRPSolver é indicado a seguir:

- *Grafo para o Subproblema RCSP*

  $G = (V, A), v_{source} = v_{sink} = 0;$
  $R = R_M = \{1, 2\};$
  $q_{a,1} = d_j, a \in A;$
  $q_{a,2} = t_{ij}, a \in A;$
  $[l_{i,1}, u_{i,1}] = [0, Q], i \in V;$
  $[l_{i,2}, u_{i,2}] = [a_i, b_i], i \in V;$

- Formulação

$$min \sum_{(i,j) \in A} c_{ij} x_{ij} \tag{6a}$$

$$\sum_{(i,j) \in \delta^-(j)} x_{ij} = 1, \quad \forall j \in N \tag{6b}$$

  Mapeamento $M(x_{ij}) = \{(i, j)\}, (i, j) \in A;$
  $L = 0, U = n.$

  Além da modelagem básica foi definido alguns componentes que fazem parte do estado da arte em algoritmos BCP, para a otimização do algoritmos, pode ser melhor entendido em [9]. São eles:

  – Packing sets e elementary sets: $P^V = E^V = \cup_{i \in V}\{\{i\}\}$
  – RCC Separator: $(\cup_{i \in V}\{(\{i\}, d_i)\}, Q)$

No grafo G definido acima, foi adicionado dois recursos principais. O recurso de número 1 representa a restrição de capacidade, quando um veículo utiliza a aresta $a = (i, j)$ o seu consumo $q_{a,1}$ para essa aresta é dado pela demanda d_j do cliente j. Além disso, para respeitar a restrição de capacidade do problema foi definido que em cada um dos vértices i o limite superior de consumo acumulado $u_{i,1} = Q$. O recurso de número 2 representa a restrição de tempo (janela), quando um veículo utiliza a aresta $a = (i, j)$ o seu consumo $q_a, 2$ para essa aresta é dado pelo tempo de deslocamento t_{ij} entre os vértices i e j. Seguindo a definição do problema foi definido como limites de consumo acumulado para o recurso de tempo $[l_{i,2}, u_{i,2}] = [a_i, b_i]$. Na Figura 1 é ilustrado um grafo para a seguinte instancia:

Na Formulação, 6a representa a função objetivo clássica que se quer minimizar e em relação ao Modelo genérico 2 representa 2a. As Restrição de grau 6b são definidos em 2b. As restrições 2c são implicitamente definidos pela função de mapeamento definida M. As restrições 2d são definidos pelo valor de L e U. Além disso as variáveis λ são implicitamente definidas como o conjunto de caminhos com recurso restrito.

Table 1: Instância Exemplo

i	d_i	a_i	b_i	s_i
0	0	0	30	0
1	3	4	8	1
2	4	1	10	1
3	10	2	6	1

Figure 1: Exemplo de grafo com restrição de recursos.

5 ANÁLISE DOS RESULTADOS

O modelo criado no VRPSolver foi testado em 2 conjuntos de instancias diferentes. O primeiro conjunto é formado pelas tradicionais instâncias de Solomon [12]. Além disso, afim de utilizar instancias do benchmark de entregas urbanas(loggiBUD) proposto pela empresa Loggi [6], foi modificado algumas instancias disponíveis no benchmark. Foram adicionadas janelas de tempo para cada um dos clientes e tempo de serviço para cada um dos clientes pertencentes a uma instancia. As instancias disponíveis no benchmark foram criadas sinteticamente com dados públicos, e simula um desfio real de uma grande empresa na etapa final da cadeia de suprimentos. O benchmark é interessante por representar grandes cidades brasileiras (PA,DF, RJ) com demandas específicas para as regiões diferentes.

Na Seção 5.1 será explicado o processo para a adaptação de instancias a partir das instancias disponíveis no loggiBUD. Na seção 5.2 são apresentados os resultados para os experimentos realizados.

5.1 Instâncias adaptadas

No repositório do benchmark loggiBUD são disponibilizados dois tipos de instancias, *delivery-instances* e *cvrp-instances*. As instancias

trabalhadas neste artigo são as do segundo tipo as quais representam o clássico problema do roteamento de veículos (CVRP). Nessas instancias são apresentados a localização do depósito, a capacidade Q dos veículos da frota, as localizações de cada entrega a ser feita, além das demandas de cada uma das entregas. O custo de deslocamento cij de um cliente i para um cliente j são calculados através de distâncias de mapas utilizando ruas reais (http://project-osrm.org), assim tem-se uma função de distância diferente da tradicional Euclidiana utilizada na maior parte dos trabalhos da literatura.

O benchmark possui 1320 instancias do tipo *cvrp-instances*, foi selecionado apenas algumas poucas instancias para adicionar janela de tempo. Foram gerados 4 tipos diferentes de instancias:

- R - Instâncias com Janelas de Tempo aleatórias;
- F30m - Instâncias com Janelas de Tempo fixas de 30 min;
- F1h - Instâncias com Janelas de Tempo fixas de 1h;
- F2h - Instâncias com Janelas de Tempo fixas de 2h;

Para cada um dos tipos foram gerados 15 instâncias, sendo 5 geradas a partir de instancias para o estado de PA, e 5 a partir de instancias para o estado de DF e 5 para RJ. Em termos de número de entregas a serem feitas as instancias rj são as maiores com mais de 1000 entregas a serem feitas por instancia, seguido pelas instancias df que variam entre 700 e 1000 por instancia, e por ultimo as instancias pa são as menores contendo entre 200 e 400 entregas por instancia.

Para construir as novas instancias foi utilizado as distâncias entre os clientes em metros disponibilizadas pelo servidor OSRM. Para definir as janelas fixas de tempo ficou definido que um veículo anda em uma velocidade constante de $30km/h$ então, $1h$ para as instancias são equivalentes a 30000 metros de deslocamento. As instâncias foram construídas com um plano de horizonte $12h$, assim para todas as instâncias $[a_0, b_0] = [0, 360000]$.

As instâncias com janelas de tempo aleatórias foram definidas assim como em [12], onde o centro da janela é definido aleatoriamente entre o intervalo $[a_0 + t_{0i}, b_0 - t_{0i} - s_i]$, a largura da janela também é definida aleatoriamente de maneira que a janela ainda respeite o intervalo. Nas instâncias com janela de tamanho fixo, o centro da janela é definido da mesma maneira, no entanto a largura passa de ser definida aleatoriamente para assumir um valor fixo.

5.2 Experimentos

Os experimentos foram realizados utilizando um computador com 16Gb de memória disponível e processador intel Core i7, 8ª geração. Além disso a implementação base utilizada é a mesma da demo disponível no site do VRPSolver (https://vrpsolver.math.u-bordeaux.fr), apenas algumas mudanças relacionadas a pré e pós-processamento de dados foram realizados.

Foi refeito o experimento realizado realizado em [9], Algumas instâncias de Solomon (cada uma com 100 entregas a serem feitas) foram testadas para comparar seus resultados com as instâncias geradas neste trabalho. Todas as instâncias testadas fazem parte do conjunto de instancias com um grande horizonte de escalonamento, o que gera soluções com mais clientes atendidos em cada rota. No experimento, foram resolvidas quase todas as instâncias dentro do tempo limite definido (1.5h), exceto a instancia R208. Todas as instâncias resolvidas levaram menos de 15 minutos para encontrarem o ótimo, o que revela que o método funciona bem. Mesmo que os experimentos tenham sido realizados em um computador pessoal instancias não resolvidas antes de [8] foram resolvidas. Na tabela 2 estão dispostos os resultados para ps experimentos.

Table 2: Experimento - Instâncias Solomon

Instância	Tempo (s)	Custo Solução
C203	20.78	588.7
C204	46.61	588.1
R202	184.30	1029.6
R203	93.84	870.8
R204	470.02	731.3
R206	527.43	875.9
R207	305.83	794.0
R208	> 5400	
R209	549.01	854.8
R210	897.46	900.5
R211	656.16	746.7
RC204	339.39	783.5
RC207	217.88	962.9
RC208	344.38	776.09

O método de geração de coluna utilizado pelo VRPSolver funciona melhor com o problema do VRPTW quando suas janelas são mais apertadas. Isso pode ser visto nesse trabalho, onde para as instancias geradas a a partir do loggiBUD nenhuma das instâncias com janelas aleatórias conseguiu ser resolvido em menos de 2 horas de processamento. Para as instâncias com janelas fixas algumas instâncias do estado PA causaram alguma dificuldade, mas no geral conseguiram ser resolvidas em menos de 1.5 horas (tempo limite definido).

Os resultados dos experimentos para as instancias estão dispostos nas tabelas 3 e 4. Pode-se observar que o método resolveu rapidamente algumas instâncias com menos de 300 entregas a serem feitas e janelas fixas. Quando o número de entregas passa de 300 o algoritmo acaba gastando bastante tempo de processamento para resolver.

Table 3: Experimento - Instâncias com Janela Fixa 30 minutos

Instância	Tempo (s)	Custo Solução
F30m-pa-n289	1288.17	808037.0
F30m-pa-n266	273.04	895959.0
F30m-pa-n354	> 5400	
F30m-pa-n324	3081.95	875398.0
F30m-pa-n308	> 5400	
F30m-df-n999	> 5400	
F30m-df-n904	> 5400	
F30m-df-n974	> 5400	
F30m-df-n835	> 5400	
F30m-df-n939	> 5400	

Table 4: Experimento - Instâncias com Janela Fixa 1 hora

Instância	Tempo (s)	Custo Solução
F1h-pa-n269	486.05	880795.0
F1h-pa-n324	3474.46	1143354.0
F1h-pa-n322	> 5400	
F1h-pa-n212	958.37	645609.0
F1h-pa-n296	778.30	607808.0
F1h-df-n780	> 5400	
F1h-df-n815	> 5400	
F1h-df-n890	> 5400	
F1h-df-n957	> 5400	
F1h-df-n993	> 5400	

Figure 2: Solução para instância - F30m-pa-n266

6 CONSIDERAÇÕES FINAIS

Como principal contribuição deste trabalho está a incorporação de janelas de tempo relacionadas a cada cliente nas instâncias urbanas reais do loggiBUD, o que implica afirmar que uma tarefa intermediária pode ser incluída neste benchmark ou mesmo ter a *Task 1* que envolve o CVRP clássico sem janelas de tempo substituída, com este modelo que incorpora uma característica importante da dinamicidade necessária para atendimento na aplicação real. A *Task 2* também é beneficiada com a consideração de janelas de tempo agregadas ao cliente.

Foram testadas dois tipos de instâncias: (1) o clássico conjunto de instâncias artificiais de Solomon [12]; (2) e as instâncias reais de cidades brasileiras do benchmark loggiBUD [6], que precisaram ser adaptadas para incorporarem janelas de tempo. Os experimentos computacionais iniciais realizados apresentam resultados promissores, uma vez que foi possível resolver instancias reais com até 300 clientes, o que deve ser reforçado com uma análise experimental mais robusta e em ambiente computacional de configuração mais potente do que o usado.

Como atividades em andamento da pesquisa, está a investigação de outros tipos de janelas de tempo, incorporando características de teoria de escalonamento, onde data de disponibilidade (*release date*) e de prazo máximo de entrega desejados ou obrigatórios (*due date* e *deadlines*) da empresa são também considerados. Também, uma meta-heurística está sendo elaborada na tentativa de se estabelecer uma estratégia híbrida que permita fornecer boas soluções para instâncias reais de maior dimensão.

Agradecimentos

Este trabalho é um dos resultados do Programa de Bolsas de Pesquisa (PBP) da empresa brasileira de cadeia logística de entregas expressas Loggi, em obtenção pelo grupo de pesquisa em Otimização, Algoritmos e Complexidade Computacional (ALGOX) do IComp/UFAM, com contribuição do pesquisador Eduardo Uchoa - UFF.

REFERENCES

[1] Leonardo José Elias Cavalcanti and Nicolas Franco Doneux. 2021. Análise de fatores determinantes na decisão de compra online: reflexões sobre o impacto da pandemia no comportamento do consumidor brasileiro. (2021).
[2] Guy Desaulniers, Oli BG Madsen, and Stefan Ropke. 2014. Chapter 5: The vehicle routing problem with time windows. In *Vehicle Routing: Problems, Methods, and Applications, Second Edition*. SIAM, 119–159.
[3] Burak Eksioglu, Arif Volkan Vural, and Arnold Reisman. 2009. The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering* 57, 4 (2009), 1472–1483.
[4] Juan Camilo Fonseca-Galindo, Gabriela de Castro Surita, José Maia Neto, Cristiano Leite de Castro, and André Paim Lemos. 2020. A Multi-Agent System for Solving the Dynamic Capacitated Vehicle Routing Problem with Stochastic Customers using Trajectory Data Mining. *arXiv preprint arXiv:2009.12691* (2020).
[5] Niklas Kohl, Jacques Desrosiers, Oli BG Madsen, Marius M Solomon, and Francois Soumis. 1999. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science* 33, 1 (1999), 101–116.
[6] Loggi. 2021. *loggiBUD: Loggi Benchmark for Urban Deliveries.*
[7] Mehdi Nasri, Imad Hafidi, and Abdelmoutalib Metrane. 2021. Multithreading parallel robust approach for the VRPTW with uncertain service and travel times. *Symmetry* 13, 1 (2021), 36.
[8] Diego Pecin, Claudio Contardo, Guy Desaulniers, and Eduardo Uchoa. 2017. New enhancements for the exact solution of the vehicle routing problem with time windows. *INFORMS Journal on Computing* 29, 3 (2017), 489–502.
[9] Artur Pessoa, Ruslan Sadykov, Eduardo Uchoa, and François Vanderbeck. 2020. A generic exact solver for vehicle routing and related problems. *Mathematical Programming* 183, 1 (2020), 483–523.
[10] Luigi Di Puglia Pugliese and Francesca Guerriero. 2013. A survey of resource constrained shortest path problems: Exact solution approaches. *Networks* 62, 3 (2013), 183–200.
[11] Stefan Røpke. 2009. PALNS-a software framework for parallel large neighborhood search. In *Metaheuristic International Conference.*
[12] Marius M Solomon. 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research* 35, 2 (1987), 254–265.
[13] Zheng Wang and Wei-Hua Lin. 2017. Incorporating travel time uncertainty into the design of service regions for delivery/pickup problems with time windows. *Expert Systems with Applications* 72 (2017), 207–220.
[14] Esam Taha Yassen, Masri Ayob, Mohd Zakree Ahmad Nazri, and Nasser R Sabar. 2017. An adaptive hybrid algorithm for vehicle routing problems with time windows. *Computers & Industrial Engineering* 113 (2017), 382–391.

Análise, Definição e Experimentação de Critérios de Teste de Software para Integração de Sistemas via API, Serviços ou Microsserviços

Igor Ribeiro Nascimento Torga
Auri Marcelo Rizzo Vincenzi
torgaigor@estudante.ufscar.br
auri@ufscar.br
Departamento de Computação - Universidade Federal de São Carlos
São Carlos, São Paulo, Brasil

ABSTRACT

Objetivo: Definir um conjunto de operadores de mutação para teste de integração de sistemas Python composto por integração de API, serviços ou microsserviços. **Método**: O trabalho consiste na definição de um conjunto operadores de mutação para teste de integração. Para atingir tal objetivo as seguintes etapas são propostas: 1) pesquisar a literatura em busca de operadores de mutação que representam defeitos de integração em diferentes linguagens de programação; 2) estudas aspectos relacionados à integração presentes na linguagem Python; 3) identificar operadores de integração já existentes que representam defeitos de integração no contexto Python; 4) propor novos operadores de integração específicos para Python que ainda não existam na literatura; 5) conduzir estudos experimentais para avaliar eficácia dos operadores definidos; 6) automatizar a aplicação do teste de mutação de interface para sistemas Python. **Resultados esperados**: um conjunto de operadores de mutação para teste de integração; um protótipo de ferramenta que viabilize o teste de mutação de interface em Python, e um conjunto de dados resultante da aplicação dos operadores de mutação nos programas Python. Nesse sentido, diferentes categorias de experimentos, entre as quais as técnicas e critérios de teste de avaliação, podem ser conduzidos com especial atenção em estudos de caso na indústria. Pretende-se coletar dados quantitativos e qualitativos enfatizando o teste de integração para apoiar o teste e a atividade de transferência de tecnologia no nível de integração para criar uma estratégia de teste voltada para sistemas compostos de integração de API, serviços ou microsserviços. Além disso, pretende-se desenvolver um protótipo de ferramenta de teste de mutação, para suportar o conjunto de operadores de mutação proposto para testes de integração em Python. **Conclusão**: Considerando o estágio atual do projeto, o estudo identificou na literatura os operadores de mutação que são representativos em defeitos de integração de sistemas desenvolvidos com base na integração via API, serviços ou microsserviços. Esse conjunto de operadores contribui para encontrar defeitos e a melhorar os testes no contexto de produtos Python.

CCS CONCEPTS

• **Software and its engineering → Software notations and tools**; **Software creation and management**; **Software verification and validation**; **Empirical software validation**;

KEYWORDS

teste de software, teste de mutação, linguagem python, automatização de teste de mutação

ACM Reference Format:
Igor Ribeiro Nascimento Torga and Auri Marcelo Rizzo Vincenzi. 2021. Análise, Definição e Experimentação de Critérios de Teste de Software para Integração de Sistemas via API, Serviços ou Microsserviços. In *Workshop Loggi'21, 8 de dezembro, 2018, São Paulo, SP.* ACM, New York, NY, USA, 6 pages. https://doi.org/xx.xxxx/xxxxxxx.xxxxxxx

1 INTRODUÇÃO

As empresas de software adotam, cada vez mais, práticas de garantia de qualidade, devido ao aumento na exigência de qualidade e aumento da complexidade de serviços disponibilizados [17]. O sucesso do desenvolvimento de um sistema de software e a sua qualidade dependem de diversos fatores, mas possuem uma ligação direta com as práticas utilizadas para verificação e validação do sistema desenvolvido, sendo o teste uma das práticas mais utilizadas [12]. Nesse sentido, o surgimento da arquitetura de microsserviços é apontado como solução para a divisão de grandes sistemas monolíticos complexos, em partes menores e menos complexas, teoricamente, contribuindo também com sua testabilidade [12, 17].

A atividade de Teste de Software, é uma atividade que independe da arquitetura apresentada no software, sendo essencial para a garantia de qualidade, visto que sua aplicabilidade auxilia na diminuição de custo para empresas [18] decorrentes de falhas encontradas após o lançamento do produto. O teste de unidade exercita o menor pedaço de software testável no aplicativo para determinar se ele se comporta conforme o esperado [9] e o teste de integração verifica as interações entre os componentes ou sistemas, de forma que as informações capturadas na interface do usuário são transmitidas para a lógica de negócios para se detectarem os defeitos [9, 21].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
WLOGGI'21, 8 de dezembro de 2021,, São Paulo, SP
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/xx.xxxx/xxxxxxx.xxxxxxx

Segundo Delamaro et al. [4], o cenário ideal seria testar o software à exaustão, mas a possibilidade de infinitas combinações de um domínio de entrada, além do tempo limitado para o desenvolvimento e entrega do produto, torna o teste exaustivo impraticável. Neste sentido, definem-se técnicas e critérios que irão auxiliar no projeto de casos de teste de forma eficaz e eficiente.

A técnica de testes baseada em defeitos utiliza a premissa de injetar defeitos conhecidos no código fonte para encontrar defeitos que não foram detectados pela equipe [5]. O critério de Teste de Mutação, um dos critérios da técnica de teste baseada em defeitos, tem o objetivo central de aplicar mutações de código, essas mutações são pequenas alterações sintáticas no código, as quais representam possíveis defeitos, garantindo testes mais eficazes ou localizando pontos fracos nos casos de teste usados para o programa ou em seções do código que raramente ou nunca são o acessadas durante a execução. O Teste de Mutação, um dos critério da técnica baseada em defeitos, é tradicionalmente empregado no teste de unidade e deve garantir o aumento da cobertura de testes.

Delamaro et al. [5] propuseram uma extensão do teste de mutação para o teste de integração. O critério denominado Mutação de Interface foi concebido por meio da modelagem de operadores de mutação que representam defeitos típicos de integração em programas C.

Para apoiar o teste de mutação e viabilizar seu uso na prática é de fundamental importância a existência de uma ferramenta que automatize sua aplicação. No caso de C, Delamaro et al. [6] propuseram a ferramenta Proteum/IM que apoia o teste de mutação de unidade e integração para programas escritos nessa linguagem. Existem outras ferramentas que apoiam o teste de mutação em outras linguagens, tais como: MuJava [16], Pit [3], Major [13] e Javalanche [19] para Java e Cosmic-Ray [1], MutMut [11], MutPy [10] e Mutatest [14] para Python. A existência das ferramentas facilita atividades de ensino, treinamento e capacitação no emprego do teste de mutação, além de contribuir para a transferência tecnológica.

Nesse contexto, este trabalho visa a realização de um estudo de como aproveitar os conceitos do critério Mutação de Interface no teste de sistemas compostos por meio da integração de APIs, serviços e/ou microsserviços escritos em Python. Pretende-se conduzir estudos experimentais utilizando-se sistemas open-source e sistemas privados, visando avaliar aspectos de custo e eficácia e dificuldade de satisfação dos operadores de mutação nesse contexto. Para validar tanto o processo quanto o protótipo do ambiente de apoio, fundamental se faz a condução de estudos experimentais para o levantamento de evidências sobre os benefícios, vantagens e desvantagens dos operadores de mutação de interface definidos. A motivação para o uso do teste de mutação é que esse se mostra um excelente modelo de defeitos, sendo constantemente utilizado para avaliar a qualidade de conjuntos de teste definidos por outros critérios de teste.

O restante deste texto está organizado da seguinte forma. Na Seção 2 são descritas as principais terminologias e conceitos básicos necessários para a compreensão deste artigo. Na Seção 3 são descritos alguns trabalhos da literatura relacionados com este artigo. Na Seção 4 é apresentado os operadores de mutação para integração e a forma como foi conduzido os operadores destinados para programas Python, e os resultados obtidos. Finalmente, na Seção 5 são apresentadas as conclusões deste trabalho, bem como as perspectivas de trabalhos futuros.

2 TERMINOLOGIA E CONCEITOS BÁSICOS

Esta seção apresenta alguns termos e conceitos básicos relacionados ao presente trabalho.

2.1 Serviços e Microsserviços

Para Carneiro e Schmelmer [2], a definição de serviço de software é um sistema que atende a uma necessidade pública. A interface de tais serviços deve, portanto, ser projetada da perspectiva do cliente, otimizando para a máxima utilidade para seu consumidor. Embora idealmente cada serviço seja diferente, pois otimiza os casos de uso de seus clientes, geralmente pode-se dizer que um serviço consiste em uma parte da funcionalidade e seu conjunto de dados.

A arquitetura de microsserviços é apontada como solução para a divisão de grandes sistemas monolíticos complexos, em partes menores e menos complexas [18]. O uso dessa arquitetura é baseado em um conhecido princípio adotado na engenharia de software, a ideia de separar e organizar uma única aplicação em componentes mais gerenciáveis, isolados e autônomos. Soldani et al. [20] definem a arquitetura de microsserviços como um serviço independente de recursos, construído em torno de uma capacidade de negócios, é executável em seu próprio processo e comunica com outros serviços por meio de mecanismos chamadas de APIs via HTTP, por exemplo.

2.2 Teste de Software e Teste de Mutação

O processo de teste de software no desenvolvimento de software envolve atividades agregadas sob o nome de Garantia de Qualidade de Software ao longo de todo o processo, sendo uma das atividades do processo de VV&T – Verificação, Validação e Teste, com o objetivo de minimizar a ocorrência de defeitos e riscos associados [4]. Dentre as técnicas de verificação e validação, o teste é uma das mais utilizadas, constituindo-se em um dos elementos para fornecer evidências da confiabilidade do software em complemento a outras atividades.

O teste de mutação, um dos critérios da técnica baseada em defeitos, deriva os requisitos de teste (mutantes) a partir dos enganos mais frequentes cometidos durante o processo de desenvolvimento de software. O teste de mutação tem como uma das suas finalidade medir o grau de adequação de um conjunto de teste. Na teoria ele oferece ao testador recursos para, por meio da aplicação do critério teste de mutação, criar e avaliar a adequação de um conjunto de teste para um determinado programa [4]. Como resultado, o testador pode melhorar a qualidade até obter um conjunto de testes adequado ao critério definido.

2.3 Teste de Mutação de Interface

Segundo Delamaro et al. [5], o teste de mutação de interface é um critério de teste que sua preocupação em assegurar a correção da integração das conexões entre duas unidades. Assim, como um teste de mutação em nível de unidade deve ser capaz de revelar grande parte dos defeitos no programa, o teste de mutação aplicado na camada de integração deve contribuir da mesma forma. Ambos tem como objetivo contribuir no refinamento dos casos de testes para cobrir e melhorar a qualidade do programa com foco em defeitos

de unidade ou de integração, respectivamente. Como resultado do estudo, obteve-se na análise que a complexidade dos operadores definidos é bastante alta dada a quantidade de mutantes gerados pelo critério de Mutação de Interface, mesmo para o teste de uma conexão formada por duas unidades simples.

2.4 Automatização de Teste de Mutação

A automação é um requisito importante na garantia de qualidade em um projeto. Proporciona uma redução significativa de esforço por parte dos colaboradores e reduz a chance de enganos decorrentes da intervenção humana. Para os testes baseados em mutação, a automação é uma tarefa essencial, se pensarmos na viabilidade de inserção de testes de forma manual em um programa para quebrar o código e garantir a qualidade dos casos de testes. Na prática, uma ferramenta que realiza a automatização de testes de mutação garante que todos operadores sejam aplicados em um programa e praticidade de uso da técnica, reduzindo tempo e custo. Atualmente, não existe nenhuma ferramenta com operadores em nível de integração para a linguagem Python, apenas em nível de unidade.

3 TRABALHOS RELACIONADOS

No trabalho de Delamaro et al. [4], os autores apresentaram um estudo, sobre as técnicas de teste funcional, estrutural e baseadas em erros. Cujo, os critérios de testes das técnicas, são fatores de comparação e avaliação como custo, eficácia e dificuldade de satisfação. Como resultado, os benefícios do trabalho são o nível de detalhamento na criação de operadores de mutação e as ferramentas geradas do trabalho, aspectos técnicos e teóricos para os critérios na camada de integração e adoção da técnica de Mutação de Interface, a identificação de possíveis estudos, iniciativas e esforços da comunidade para a automação de critérios, a desvantagem observada é a baixa quantidade de programas na aplicabilidade das ferramentas no estudo, mas também existe a possibilidade de um custo alto nos testes de mutação em nível de integração.

Derezinka e Halas [7] apresentaram diversos operadores de mutação para programas Python, utilizando-se de estudos de testes de mutações em outras linguagens de programação. Assim, surgiu a necessidade de validar operadores com sintaxe diferente para este cenário. Como resultados, apresentou-se a minimização de custos na aplicabilidade enfrentados pelo Teste de Mutação, um score de mutação para cada operador e mostra-se que apenas uma pequena porcentagem dos mutantes gerados eram incompetentes. Portanto, o autor afirma que se o operador está criando mutantes válidos além dos incompetentes, e esses mutantes poderiam ser usados para verificar características importantes da linguagem, então poderia ser benéfico criar um ambiente adicional para a geração em tempo de execução deste tipo de mutante.

O estudo de Derezinska e Halas [8] tem como foco uma melhoria para o processo de mutação de operadores em Python, realizando o refinamento dos operadores já existentes na ferramenta MutPy para Testes de Mutação em nível de unidade. Assim é possível que com critérios no cenário da linguagem Python, a construção de um comparativo nos operadores fornecidos pela Python AST. Os resultados do estudo garantem um melhor desempenho dos operadores e a redução da mutação implementada no MutPy, no contexto de geração de mutantes com base em modificações de uma árvore de sintaxe abstrata e execução de mutantes com testes.

Delamaro et al. [5] propuseram um estudo denominado como critério de teste de Mutação de Interface com o objetivo de criar operadores que contribuam na melhoria dos testes, e então medir os benefícios da adesão destes critérios em programas. Como resultado, mostram-se benefícios como, a melhoria do conjunto de testes, a melhoria da qualidade do programa, a diferenciação de estudos em um conjunto de operadores de mutação definido, os quais representam defeitos relacionados à integração de unidades e não às unidades isoladas. Contudo, uma desvantagem do estudo, segundo os autores, é que o estudo apresenta uma baixa quantidade de programas para validar os operadores, sendo uma tarefa futura a execução extensão do estudo de caso. No estudo obteve-se a definição de 2 grupos de operados, no Grupo I é definido como Grupo de Operadores na função chamadora, e são 24 operadores nesse grupo, estes operadores são projetados para causar mudanças nos valores que estão entrando e saindo da função, no Grupo II é denominado como Grupo de Operadores dentro da função chamada, e são 9 operadores, nesse grupo os operadores são aplicados nas instruções onde uma função chama outra, ou seja são mutações nos argumentos de chamada. Pode ser visto os operadores de ambos os grupos na Tabela 1.

No trabalho de Kim et al. [15] foi realizado um estudo para construir e avaliar operadores de mutação para linguagem Java que contribua na detecção de defeitos relacionados com as características especiais de Orientação a Objetos (OO), pois só existiam operadores de mutação em nível de unidade para programas Java. No estudo obteve-se um conjunto de 13 operadores que visa a contribuir com a qualidade do código em Java. Os operadores Java considerados estão listados na Tabela 1.

Observou-se que esses trabalhos, tem como objetivo a criação de operadores de mutação, porém não exploram propriamente a questão dos testes numa arquitetura baseada em APIs, serviços e microsserviços, percebe-se a falta de trabalhos para testes nesse nível e um modelo de defeitos adequados para a avaliação dos critérios existentes. Nesse sentido, o presente projeto visa definir um conjunto de operadores de mutação para teste de integração de sistemas Python compostos por API, serviços e microsserviços.

4 DEFINIÇÃO DE OPERADORES DE MUTAÇÃO PARA TESTE DE MICROSSERVIÇOS

A Tabela 1 apresenta a lista dos operadores de mutação de interface ou voltados para o teste de integração definidos para as linguagens C e Java, e os que possuem um "X" são os que possuem alguma correspondência com Python. O método aplicado na descoberta de operadores que servem para Python é revisando literaturas que tratam de testes de mutação, verificando o que já existe para esta linguagem, mapeando operadores de outras linguagens de programação e validando ferramentas já existentes. Como já existem ferramentas de teste de mutação para unidade, a leitura é apenas voltada para trabalhos em nível de integração. Como não existe uma ferramenta de apoio ao teste de mutação de interface para Python, um protótipo deve ser desenvolvido para apoiar a aplicação dos operadores definidos.

Table 1: Operadores encontrados na literatura que estão sendo validados para programas Python

Operadores	Linguagens		
	Java	C	Python
DirVarRepPar	-	X	X
DirVarRepGlo	-	X	X
DirVarRepLoc	-	X	X
DirVarRepExt	-	X	X
DirVarRepCon	-	X	X
DirVarRepReq	-	X	X
IndVarRepPar	-	X	X
IndVarRepGlo	-	X	X
IndVarRepLoc	-	X	X
IndVarRepExt	-	X	X
IndVarRepCon	-	X	X
IndVarRepReq	-	X	X
DirVarIncDec	-	-	-
IndVarIncDec	-	-	-
DirVarAriNeg	-	-	-
IndVarAriNeg	-	-	-
DirVarLogNeg	-	-	-
IndVarLogNeg	-	-	-
DirVarBitNeg	-	-	-
IndVarBitNeg	-	-	-
RetStaDel	-	-	-
RetStaRep	-	-	-
CovAllNod	-	-	-
CovAllEdg	-	-	-
ArgBitNeg	-	-	-
ArgIncDec	-	-	-
FunCalDel	-	-	-
ArgRepReq	-	-	-
ArgStcAli	-	-	-
ArgStcDif	-	-	-
ArgDel	-	-	-
ArgAriNeg	-	-	-
ArgLogNeg	-	-	-
CRT	X	X	-
ICE	X	X	-
POC	X	X	-
VMR	X	X	-
AOC	X	X	-
AND	X	X	-
OMR	X	X	-
HFR	X	X	-
HFA	X	X	-
AMC	X	X	-
SMC	X	X	-
EHR	X	X	-
EHC	X	X	-

4.1 Exemplos de Uso

A seguir, para cada um dois operadores que possuem alguma relação com a linguagem Python, é apresentado um pequeno trecho de código com um exemplo da mutação realizada.

Grupo de operadores na função chamadora

Grupo de operadores na função chamadora, estes operadores são projetados para causar mudanças nos valores que estão entrando ou saindo da função. Para aplicar estes operadores, dada uma conexão entre F–G, onde F chama G, é necessário identificar o local de onde a função G foi chamada, requer um mecanismo para habilitar ou desabilitar a mutação, dependendo se G é chamado de F ou não. Este mecanismo consiste em uma variável de controle que deve ser definida quando F chama G, habilitando a mutação apenas se o teste passar pelo ponto onde função f chama a função G, do contrário, a mutação em G não é habilitada pois não está se testando a conexão desejada.

```
# Programa principal
def percorrer_sequencia(a):
    numero = 0
    for numero in range(a, 10, 2):
        somar_numero(a)

def somar_numero(a):
    return a+1

# Mutante 1
def percorrer_sequencia(a):
    numero = 0
    for numero in range(IF_MUTA(numero, a), 10, 2):
        somar_numero(numero)

def somar_numero(a):
    return a+1
```

Código 1: Código exemplo do Operador de Mutação: dir_var_rep_const

```
# Programa principal
dia = []
ano = [10, 20]

def verificar_data(i):
    mes = [2, 10]
    if i == 3 and 10 in mes:
        i = i + 10


# Mutante 1
dia = []
ano = [10, 20]

def verificar_data(i):
    mes = [2, 10]
    if i == 3 and IF_MUTA(20 in ano, 10 in mes):
        i = i + 10
```

Código 2: Código exemplo do Operador de Mutação: dir_var_rep_ext

```
# Programa principal
lista = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

def somar_numeros(i, lista):
    global j
    j = 3
    for a in lista:
        i = i + lista[j]

# Mutante 1
lista = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

def somar_numeros(i, lista):
    global j
    j = 3
    for a in lista:
        i = i + IF_MUTA(a, lista[j])
```

Código 3: Código exemplo do Operador de Mutação: dir_var_rep_glob

Grupo de operadores aplicados na função chamada

Grupo de operadores na função chamada, estes operadores dada uma conexão A–B, esses operadores são aplicados as instruções onde A chama B e para argumentos de chamada. Quando um argumento é uma expressão, os operadores neste grupo são aplicados a toda a expressão, não a objetos dentro da expressão como variáveis ou constantes. Os benefícios que estes operadores forçam a

```
# Programa principal
def contar_valor(i):
    x = y = z = 0
    for x in range(i):
        y = y + 30
        z = z / y
    return z

# Mutante 1
def contar_valor(i):
    x = y = z = 0
    for x in range(IF_MUTA(x, i)):
        y = y + 30
        z = z / y
    return z
```

Código 4: Código exemplo do Operador de Mutação: dir_var_rep_loc

```
# Programa principal
a = 0

def juntar_valor(i, vet):
    for x in range(i):
        a += vet[x]
    return a

# Mutante 1
a = 0

def juntar_valor(i, vet):
    for x in range(i):
        IF_MUTA(DEVOLVE_VAR(a = a + i), a = a + vet[x])
        return a
```

Código 5: Código exemplo do Operador de Mutação: dir_var_rep_par

```
# Programa principal
import sys

max = sys.maxsize

def contar_valor(i):
   x, y = 0
   for (x == 1) in range(i):
       y = i + 100

# Mutante 1
import sys

max = sys.maxsize

def contar_valor(i):
   x, y = 0
   for (x == 1) in range(i):
       IF_MUTA(DEVOLVE_VAR(y=max + 30), y=i + 30)
```

Código 6: Código exemplo do Operador de Mutação: dir_var_rep_req

seleção dos casos de testes que mostram que cada parâmetro tem uma função distinta na interface de função. A aplicação do Grupo operadores para cada conexão em um sistema garante a cobertura de todas as chamadas do programa, uma exigência mínima para qualquer critério inter-procedural.

5 CONCLUSÃO

O estudo tenta identificar operadores de mutação de interface ou destinados ao teste de integração já existentes na literatura e mapeá-los para o contexto de sistemas desenvolvidos em Python com base

```
# Programa principal
def aumentar_vetor(lista):
    item = 1
    for item in range(len(lista)):
        lista.append(100)

# Mutante 1
def aumentar_vetor(lista):
    item = 1
    for item in range(len(lista)):
        IF_MUTA(lista.append(5), lista.append(100))
```

Código 7: Código exemplo do Operador de Mutação: ind_var_req_const

```
# Programa principal
numero = 10

def aumentar_vetor(lista):
    item = 1
    for item in range(item, len(lista)):
        lista.append(100)

# Mutante 1

def aumentar_vetor(lista):
    item = 1
    for item in range(IF_MUTA(item, numero), len(lista)):
        lista.append(100)
```

Código 8: Código exemplo do Operador de Mutação:ind_var_req_ext

```
# Programa principal
numero = 10

def aumentar_vetor(lista):
    item = 1
    numero += item
    for item in range(item, len(lista)):
        lista.append(100)

# Mutante 1
numero = 10

def aumentar_vetor(lista):
    item = 1
    numero += item
    for item in range(IF_MUTA(numero, item), len(lista)):
        lista.append(100)
```

Código 9: Código exemplo do Operador de Mutação: ind_var_req_glob

na integração via API, serviços ou microsserviços. A definição de operadores de mutação de interface visa a modelar defeitos que se manifestam além de um único módulo e ajuda a contribuir para melhorar a qualidade do conjunto de teste, assim contribuindo com a qualidade geral do sistema.

O experimento surgiu após a realização do mapeamento de operadores que fazem sentido em programas Python, então foram mapeados 34 operadores da literatura, sendo validados 16 que podem ser utilizados em Python, 2 que não fazem sentido para a sintaxe Python, e 16 operadores que não se chegou a uma conclusão.

Dentre os próximos passos para o projeto em desenvolvimento estão a definição de um protótipo que viabilize a implementação e uso dos operadores definidos no contexto de sistemas Python e a condução de estudos experimentais visando a avaliar os aspectos de custo, eficácia e dificuldade de satisfação dos operadores definidos.

```
# Programa principal
meses = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

def troca_mes(i):
    mes_atual = 2
    for item in range(i):
        if item == 1:
            meses = 100

# Mutante 1
meses = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

def troca_mes(i):
    mes_atual = 2
    for item in range(i):
        if item == 1:
            IF_MUTA(mes_atual=100, meses = 100)
```

Código 10: Código exemplo do Operador de Mutação: ind_var_req_loc

```
# Programa principal
def aumentar_vetor(i, lista):
    item = 1
    for item in range(item, len(lista)):
        lista.append(100)

# Mutante 1
def aumentar_vetor(i, lista):
    item = 1
    for item in range(IF_MUTA(i, item), len(lista)):
        lista.append(100)
```

Código 11: Código exemplo do Operador de Mutação: ind_var_req_par

```
# Programa principal
MAX = 9999

def aumentar_valor(i, vet):
    x = 1
    for item in range(x, i):
        vet[item] = 100

# Mutante 1
MAX = 9999

def aumentar_valor (i, vet):
    x = 1
    for item in range(IF_MUTA(MAX, x), i):
        vet[item] = 100
```

Código 12: Código exemplo do Operador de Mutação: ind_var_req_req

Além disso, novos operadores podem ser definidos visando a explorar características especificas da linguagem Python, ainda não capturados por operadores definidos para as linguagens C e Java.

Além disso, como existem poucos estudos nessa área, a ideia é incentivar as comunidades e empresas a utilizarem a ferramenta, mostrando-lhes os benefícios da aplicação dos operadores de mutação de interface para melhoria de conjuntos de teste.

ACKNOWLEDGMENTS

Loggi Tecnologia pelo apoio financeiro.

REFERENCES

[1] Austin Bingham. 2021. Cosmic Ray: mutation testing for Python. (Nov. 2021). https://github.com/sixty-north/cosmic-ray

[2] Cloves Carneiro Jr and Tim Schmelmer. 2016. *Microservices from day one: build robust and scalable software from the start.* Apress.

[3] Henry Coles. 2015. PITest: real world mutation testing. Disponível em: http://pitest.org/. Acesso em: 04/07/2016. bibtex*[howpublished=Página Web].

[4] Marcio Delamaro, Mario Jino, and José Maldonado. 2016. *Introdução ao teste de software* (2 ed.). Elsevier Brasil.

[5] Marcio Eduardo Delamaro, JC Maidonado, and Aditya P. Mathur. 2001. Interface mutation: An approach for integration testing. *IEEE transactions on software engineering* 27, 3 (2001), 228–247.

[6] M. E. Delamaro, J. C. Maldonado, and A. M. R. Vincenzi. 2000. Proteum/IM 2.0: An Integrated Mutation Testing Environment. In *Mutation 2000 Symposium*. Kluwer Academic Publishers, 91–101. event-place: San Jose, CA.

[7] Anna Derezińska and Konrad Hałas. 2014. Analysis of mutation operators for the python language. In *Proceedings of the Ninth International Conference on Dependability and Complex Systems DepCoS-RELCOMEX. June 30–July 4, 2014, Brunów, Poland*. Springer, 155–164.

[8] Anna Derezinska and Konrad Hałas. 2015. Improving mutation testing process of python programs. In *Software Engineering in Intelligent Systems*. Springer, 233–242.

[9] M Fowler. 2014. Testing strategies in a Microservice Architecture. *martinfowler. com.[Online]. Available: https://martinfowler. com/articles/microservice-testing* (2014).

[10] Philipp Hossner, Konrad Hałas, Steven Myint, and Andreas Mueller. 2021. MutPy: a mutation testing tool for Python 3.x source code. https://github.com/mutpy/mutpy

[11] Anders Hovmöller. 2021. Mutmut: a Python mutation testing system. https://hackernoon.com/mutmut-a-python-mutation-testing-system-9b9639356c78

[12] RF Jorge, AMR Vincenzi, ME Delamaro, and JC Maldonado. 2002. Relatório dos Operadores de Mutação implementados nas ferramentas Proteum e PROTEUM/IM. *ICMC-Usp, São Carlos, SP* (2002).

[13] René Just, Franz Schweiggert, and Gregory M. Kapfhammer. 2011. MAJOR: An efficient and extensible tool for mutation analysis in a Java compiler. In *Proceedings of the International Conference on Automated Software Engineering (ASE)*. 612–615.

[14] Evan Kepner. 2011. Mutatest: Python mutation testing. https://github.com/EvanKepner/mutatest

[15] Sunwoo Kim, John A Clark, and John A McDermid. 2000. Class mutation: Mutation testing for object-oriented programs. In *Proc. Net. ObjectDays*. Net. Objects Erfurt, Germany, 9–12.

[16] Y.-S. Ma, J. Offutt, and Y. R. Kwon. 2005. MuJava: an automated class mutation system: Research Articles. *STVR – Software Testing, Verification and Reliability* 15, 2 (2005), 97–133. https://doi.org/10.1002/stvr.v15:2 bibtex*[location=Chichester, UK, UK;publisher=John Wiley and Sons Ltd.].

[17] Dmitrii Savchenko. 2019. Testing microservice applications. (2019).

[18] Dmitrii Savchenko, Gleb Radchenko, Timo Hynninen, and Ossi Taipale. 2018. Microservice test process: Design and implementation. *INTERNATIONAL JOURNAL ON INFORMATION TECHNOLOGIES AND SECURITY* 10, 3 (2018), 13–24.

[19] David Schuler and Andreas Zeller. 2009. Javalanche: Efficient Mutation Testing for Java. In *Proceedings of the the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering (ESEC/FSE '09, Vol. 1)*. ACM, 297–298. https://doi.org/10.1145/1595696.1595750 bibtex*[acmid=1595750;numpages=2] event-place: Amsterdam, The Netherlands.

[20] Jacopo Soldani, Damian Andrew Tamburri, and Willem-Jan Van Den Heuvel. 2018. The pains and gains of microservices: A systematic grey literature review. *Journal of Systems and Software* 146 (2018), 215–232.

[21] Andreas Spillner and Tilo Linz. 2021. *Software Testing Foundations: A Study Guide for the Certified Tester Exam-Foundation Level-ISTQB® Compliant.* Dpunkt. verlag.

Otimização da satisfação do cliente na entrega de pacotes online utilizando demandas estocásticas

Beatriz Cabral Santalucia
beatriz.santalucia@usp.br
Escola de Artes, Ciências e Humanidades – Universidade de São Paulo (EACH-USP)
São Paulo, São Paulo, Brasil

Alexandre da Silva Freire
asfreire@usp.br
Escola de Artes, Ciências e Humanidades – Universidade de São Paulo (EACH-USP)
São Paulo, São Paulo, Brasil

ABSTRACT

Neste artigo está descrito um modelo para otimizar a satisfação dos clientes com a entrega de pacotes online. Por online, entende-se pacotes que o centro de expedição logístico (local onde os pacotes de entrega são armazenados antes da entrega ao consumidor) ainda não tem conhecimento até o momento do recebimento do pacote. Nessa proposta estamos realizando a utilização da informação das demandas estocásticas para realizar o roteamento a fim de entregar o pacote para o cliente o mais rápido possível, considerando um limite de custos operacionais, maximizando assim a satisfação do cliente.

ACM Reference Format:
Beatriz Cabral Santalucia and Alexandre da Silva Freire. 2021. Otimização da satisfação do cliente na entrega de pacotes online utilizando demandas estocásticas. In *Proceedings of Workshop de Resultados PBP-Loggi.* ACM, New York, NY, USA, 5 pages.

1 INTRODUÇÃO

Nos últimos anos, com o aumento da utilização da internet e a digitalização contínua dos serviços, o comércio eletrônico vem ganhando espaço na estrutura de varejo global. As projeções para os próximos anos indicam que o número de vendas online continuará em crescimento. [7]

Com contínuo aumento das vendas pela internet, se faz necessário também tornar mais eficiente as formas de entrega dos produtos, barateando o serviço para o consumidor final e entregando em menor prazo.

O Problema de Roteamento de Veículos é um dos problemas mais estudados em otimização combinatória. Informalmente, ele consiste em planejar rotas para veículos, que partem de um ou mais pontos determinados.[6] Ele e algumas das suas variantes, são objeto de estudo para atender a logística de entrega de última milha.

Na logística do transporte de mercadorias, a última milha trata-se do transporte no qual a mercadoria sai do centro de expedição para o destino final, sendo importante porque está relacionada à determinação da qualidade e à satisfação dos clientes.

Conforme podemos ver na figura 1, o modelo de entrega logístico é constituido por 3 fases, a entrega de primeira milha (First Mile) que é a chegada do produto ao centro de distribuição. A entrega de milha intermediária (Middle Mile) que é a entrega do produto até o centro de expedição. E por fim, como explicado anteriormente a entrega de última milha.

Figure 1: Modelo de Distribuição Logística

2 OBJETIVO

Na entrega de última milha, para atender rápido e com eficiência é necessário melhorar as tecnologias disponíveis para as tomadas de decisão.

No momento de triagem dos pacotes dentro do centro de distribuição, são executados alguns processos que separam os pacotes, e hoje já são utilizadas algumas abordagens para otimização da entrega.

Com o objetivo de melhorar os processos existentes iremos propor novas abordagens deste problema, utilizando os estudos de problemas estocásticos, programação linear inteira e abordagens heuristicas.

Pensando no objetivo de satisfazer mais os clientes na entrega, a nossa proposta está pautada em realizar entregas mais rápidas, assumindo que quanto mais rápido o cliente receber a entrega mais satisfeito ele estará.

3 ATIVIDADES DESENVOLVIDAS

Ao longo deste projeto, houve a execução das atividades que foram propostas no início da pesquisa, para o programa de Bolsas de Pesquisa da Loggi (PBP-LOGGI).

Foi realizada a criação de um protocolo de condução da revisão sistemática, sucedido pelas execuções do mesmo e avaliação dos resultados encontrados. Em um primeiro momento o objetivo era explorar didáticamente a atividade de revisão sistemática.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Workshop de Resultados PBP-Loggi, 8 de Dezembro de 2021, São Paulo, SP - Brasil
© 2021 Association for Computing Machinery.

Posteriormente houve a definição de que seria executado um procedimento de revisão bibliográfica exploratória, reunindo as literaturas necessárias para o desenvolvimento do trabalho.

No decorrer da pesquisa houve o estudo dos problemas descritos no LoggiBud [9] e o início da definição formal dos mesmos. Ao longo a realização dos estudos, o escopo do projeto foi sendo delimitado e acabou-se chegando em uma proposta que se alinha mais com os objetivos propostos da pesquisa.

Posteriormente a revisão bibliográfica, houve a realização de uma experimentação de desenvolvimento, de um modelo já existente, que resolve o problema de roteirização de veículos capacitados. O desenvolvimento foi realizado na linguagem Python, utilizando a biblioteca Gurobi, que é um solucionador de otimização comercial para programação linear inteira.

A intenção desse desenvolvimento era a familiarização com a aplicação do solucionador dentro de um modelo existente, para que no futuro seja possível aplicar no modelo que está sendo definido formalmente. E também aprofundar os estudos sobre programação linear inteira.

Após o desenvolvimento, os estudos sobre programação linear inteira, programação estocástica e sobre alguns algoritmos de otimização combinatória progrediram. A intenção era encontrar referências para desenvolver a pesquisa de forma satisfatória.

Durante essa fase foi feita a análise de alguns artigos produzidos sobre o tema, com a intenção de buscar inspiração para a definição do escopo da pesquisa.

Houve um avanço na definição do escopo, direcionando o mesmo para se alinhar ao objetivo de realizar as entregas de forma eficiente e satisfazer os clientes, utilizando o conceito de que quanto antes uma entrega for realizada, mais satisfeito estará o cliente.

A ideia proposta aqui não está centralizada na minimização de distância a ser percorrida para a rota, mas sim em maximizar a satisfação do cliente realizando a entrega no menor tempo possível de forma que também seja operacionalmente viável para a empresa.

4 PROPOSTA DE TRABALHO

Para garantir a satisfação do cliente na entrega, a proposta traz um algoritmo que tem por objetivo aumentar a satisfação dos clientes.

Ele consiste em maximizar a satisfação do cliente na entrega, utilizando o critério do tempo que será gasto na realização da entrega do pacote.

Nossa proposta considera que os custos operacionais devem ser controlados, a fim de que o negócio seja operacionalmente viável, porém o objetivo é maximizar a satisfação do cliente.

Dessa forma estamos buscando garantir que as entregas sejam concluídas o mais rápido possível, ao invés de minimizar a soma total das distâncias das rotas e sem deixar de considerar os custos operacionais, que devem estar dentro de limites pré-estabelecidos.

No contexto considerado neste trabalho, o custo operacional fica a cargo dos colaboradores, que são micro-empreendedores independentes, que prestam o serviço de entrega e são remunerados por entrega realizada. Portanto, quanto menor for a quilometragem rodada por entrega, menor o custo operacional para o colaborador.

Minimizar a soma das distâncias das rotas não garante que cada rota terá um custo operacional interessante para o colaborador. Pode ocorrer que, para encontrar uma roteirização com distância total mínima, algumas rotas fiquem excessivamente custosas, ou seja, que elas tenham uma alta quilometragem por entrega). Portanto, em algumas situações, alguns colaboradores potencialmente podem ser penalizados.

Na abordagem que estamos propondo, impõem-se um limite de quilometragem por entrega que deve ser respeitado por cada rota.

Com base nas entregas realizadas no passado, pode-se estimar as demandas dos clientes no dia atual. Como o processo de entrega se repete ao longo do tempo, uma abordagem estocástica é adequada, uma vez que em tal abordagem otimiza-se em função não das demandas médias, mas sim em função das demandas de cada cenário considerado e da probabilidade de tal cenário ocorrer.

Assim, otimiza-se a expectativa de ganhos na função objetivo baseado no fato de que o processo se repetirá ao longo do tempo.

4.1 Contexto da realização das entregas

Um veículo de entrega tem uma capacidade C. Baseado no contexto real, considerando apenas entregas de pequeno porte, realizadas em motocicletas, assumiremos que C é um número pequeno (em torno de 13 pacotes).

Figure 2: Fluxo de entrega dos pacotes

Podemos assumir que o fluxo de recebimento e entrega dos pacotes, representado na figura 2, se dá da seguinte forma:

- Todos os pacotes chegam em um centro de expedição e são colocados num equipamento de separação denominado *sorter*.
- Os pacotes chegam a todo momento e de forma *online*. Ou seja, só se tem conhecimento do pacote no instante que ele chega.
- Ao passar pelo equipamento de separação, o pacote será separado e armazenado em uma bolsa que aqui estamos denominando *bag*.
- Quando a quantidade de pacotes colocados em um *bag* atinge a capacidade C o mesmo é fechado e encaminhado para

o pátio, com a finalidade de que um entregador realize as entregas correspondentes.

- Consideramos que depois de fechado, cada *bag* fique um certo tempo t no pátio até que o entregador saia para fazer as respectivas entregas.

O tempo que um pacote demora para chegar ao cliente é composto por:

(1) Tempo que ele fica dentro do *bag* até que o mesmo seja fechado.

(2) O tempo que o respectivo *bag* fica no pátio até ele ser encaminhado a um entregador.

(3) O tempo que o entregador demora para levar o pacote do depósito ao cliente.

Consideramos que o tempo (2) é fixo, embora na prática possa haver variações neste tempo. O objetivo é minimizar os tempos (1) e (3).

5 ALGORITMO

A nossa proposta constitui em um método que é dividido em duas fases.

5.1 Primeira Fase

Com base em dados de entregas realizadas no passado, é dada como entrada as demandas de cada cliente em h dias diferentes.

Figure 3: Mapa de entregas

O mapa na figura 3, em que estão localizados os clientes, representados por pontos de latitude e longitude na cor azul, será discretizado em $n \times m$ *células*. Ou seja, cada ponto no mapa está contido em uma coordenada (i, j), com $1 \leq i \leq n$ e $1 \leq j \leq m$.

Construímos um vetor D tridimensional em $R^{n \times m \times h}$, em que D_{ijk} contém a soma das demandas dos pontos na coordenada (i,j) no dia k.

Duas coordenadas (i,j) e (i', j') são vizinhas se $|i - i'| \leq 1$ ou $|j - j'| \leq 1$.

Figure 4: Mapa de entregas discretizado na matriz de células

Uma região é um conjunto de coordenadas, uma região R é válida se:

(a) Ela é contígua. Ou seja, se para cada par de coordenadas $c1$ e $c2$ de R, é possível chegar de $c1$ a $c2$ passando somente por coordenadas vizinhas que pertencem a R.

(b) Ela possui um custo operacional aceitável. Ou seja, pode-se estimar que, para qualquer subconjunto de C pontos sorteados dentro de R, o circuito hamiltoniano mínimo que envolve tais C pontos mais o depósito central, possui quilometragem por entrega dentro de um limite pré-estabelecido α. Nossa abordagem não garante que tal condição seja sempre satisfeita. O que propomos é verificar tal restrição por amostragem. Ou seja, sortear um certo número β de vezes, C pontos dentro de R e rodar um TSP (Algoritmo *Travelling Salesman Problem*, o problema do caixeiro viajante [14]) dentro de um *solver* (software para programação matemática), para os pontos em questão. Se o número de vezes que o *TSP solver* encontrar uma solução com custo menor ou igual a α for pelo menos um certo parâmetro λ, a região R é considerada com custo operacional aceitável.

(c) A restrição (b) é aplicada a w cenários possíveis. Um cenário corresponde a um possível dia de entregas. Poderia-se, por exemplo, considerar, para cada célula, as seguintes 5 possíbilidades em um dia de entrega:

- Demanda média;
- Demanda um pouco acima da média;
- Demanda muito acima da média;
- Demanda um pouco abaixo da média;
- Demanda muito abaixo da média.

Assim, teríamos 5^{n*m} cenários diferentes. Como se trata de um número exponencial de cenários, seria computacionalmente impraticável considerar todos esses cenários. Em nossa proposta, consideramos apenas um conjunto relativamente pequeno de $w = h$ cenários, que correspondem aos dias de entregas realizadas no passado, que são dadas como entrada

do algoritmo. Para que R seja válida, a restrição descrita em (b) deve ser satisfeita em pelo menos γ cenários diferentes.

Figure 5: Mapa de entregas, com a matriz de células e as regiões destacadas

As células que não são pontos dentro da cidade são ignoradas, por não serem destinos de entrega. Porém, por simplicidade estamos considerando a matriz gerada, com todas as células válidas.

Atrelado a cada região válida R, temos um ganho em termos de satisfação dos clientes localizados dentro de R. A cada região, será posicionado em um *bag* específico no *sorter*. Consequentemente, o tempo (1) é estimado em função da demanda total da região e o tempo (3) é estimado a partir das amostras consideradas para verificar que a região R tem custo operacional aceitável. Somando esses dois tempos ao tempo fixo (2), chega-se a uma estimativa de quanto tempo levará para que os pacotes sejam entregues dentro da região R.

Definimos f faixas de satisfação, baseadas no tempo que os pacotes demoram a serem entregues, e definimos o ganho atrelado à região R em função da faixa à qual a região se enquadra e à quantidade de clientes dentro de R.

Podemos classificar os clientes nas faixas de satisfação:

- Muito Satisfeito;
- Satisfeito;
- Pouco Satisfeito;
- Insatisfeito;
- Muito Insatisfeito.

Denotamos como $g(R)$ o ganho, em termos de satisfação dos clientes, associado à região R. Em $g(R)$ considera-se o ganho em cada um dos h cenários, multiplicado pela probabilidade P do respectivo cenário ocorrer.

Otimiza-se, portanto, uma função estocástica, o que geralmente é melhor do que considerar apenas a demanda média de cada célula.

Propomos um modelo de programação linear inteira em que associamos uma variável binária x_R a cada região válida R, sendo que $x_R = 1$ indica que a região R foi escolhida, caso contrário $x_R = 0$.

O modelo garante que as células são particionadas em regiões válidas, de forma a maximizar a satisfação dos clientes:

$$max \sum_{R \in \mathcal{R}} g(R) x_R \tag{1}$$

$$\sum_{R \in \mathcal{R}(c)} x_R = 1, para\ toda\ célula\ c \tag{2}$$

$$x_R \in \{0, 1\} \tag{3}$$

onde $\mathcal{R}$ é o conjunto de todas as regiões válidas e $\mathcal{R}(c)$ é o conjunto de todas as regiões válidas que contêm a célula c.

Devido ao número exponencial de variáveis, pretende-se resolver o modelo proposto através da técnica de geração de colunas, na qual inicia-se a resolução com um número pequeno de variáveis e à medida em que *restricted master LP* (programa linear mestre restrito, ou seja, o PL que contém apenas um subconjunto das variáveis), é resolvido, novas colunas são geradas com base nos custos reduzidos obtidos através da solução atual.

Inicialmente, propomos considerar apenas as regiões retangulares. Assim, o número de regiões passa a ser polinomial, viabilizando a resolução do problema.

Um possível próximo passo seria propor uma heurística para gerar *colunas* (variáveis) correspondentes a regiões não retangulares.

5.2 Segunda Fase

Uma vez definidas as regiões válidas que particionam o conjunto de células, de forma a maximizar a satisfação esperada dos clientes, a cada vez que chega um novo pacote no sorter, deve-se decidir em um curto espaço de tempo para qual bag o pacote deve ser encaminhado.

A decisão natural é simplesmente decidir com base na região à qual o pacote pertence.

Porém, dependendo da configuração atual dos bags, pode ser mais vantajoso encaminhar um pacote para um bag que estava reservado a uma região que não inclui o ponto em que o pacote deve ser entregue.

Para tomar essa decisão, além da maximização da satisfação dos clientes, deve-se levar em conta a restrição de custo operacional. Até o momento, ainda não temos uma proposta para a segunda fase.

Da forma como abordamos o problema, buscando uma solução estocástica, espera-se que manter a decisão da primeira fase na segunda fase nos leva ao ganho esperado máximo ao longo das múltiplas execuções do algoritmo, embora não haja garantia de que todos os dias os ganhos serão máximos.

Por isso, pretendemos propor alguma heurística para a segunda fase que não faça mudanças muito drásticas no planejamento realizado na primeira fase.

6 TRABALHO EM ANDAMENTO

O desenvolvimento do algoritmo para a fase um está sendo realizado, também está sendo definido quais são os critérios que utilizaremos para os testes, decidindo sobre quais tamanhos linhas e colunas melhor se encaixam para a solução da primeira fase.

Para a construção do mapa discretizado estamos fazendo uso das instâncias ficticias geradas pelo "Loggi Benchmark for Urban Deliveries" (LoggiBUD), [9], que entregam pontos de latitude e longitude de entregas que refletem a realidade de entrega daquelas regiões, com base em dados históricos.

Ainda temos questões em aberto, como por exemplo, em relação a definição dos tempos que compõem o modelo. Para a composição do tempo (2) seria necessário definir quais os horários de funcionamento da operação logística podemos considerar para que a otimização seja o mais realista possível. E também, caso seja definido que existem intervalos na operação de um dia para o outro, que destinos terão os pacotes que não foram entregues a tempo do fechamento da operação.

Conforme o desenvolvimento é executado, as definições dos demais parametros que compõem a nossa solução, são realizadas, de forma a encontrar os que mais se encaixam na otimização.

7 TRABALHOS FUTUROS

De maneira a enriquecer a nossa proposta, os próximos passos consistem em aprofundar os estudos nas metodologias de satisfação dos clientes, como por exemplo a "Net Promoter Score" [4], com o objetivo de estruturar de melhor forma as faixas de satisfação dos clientes.

Conforme mencionado na fase um do algoritmo, também poderíamos propor uma heurística para gerar regiões não retangulares. Essa próxima heuristica poderia ser dividida em duas etapas. Em um primeiro momento vamos gerar regiões contíguas não retangulares que não compartilham células.

Uma segunda etapa possível seria gerar regiões contíguas não retangulares, que podem compartilhar células, desta forma caso um pacote seja destinado a uma região que não tem uma demanda muito recorrente, ainda é possível encaixar ele em uma região que efetuará a entrega em um tempo que atenderá melhor o critério de satisfação.

Futuramente também pretendemos criar um critério de tamanho para o *bag* de modo que ele possa atingir um tamanho mínimo para fechamento, e o tamanho máximo, que é correspondente a capacidade do veículo.

Variar os tamanhos de *bag* é vantajoso pois flexibiliza o processo de formação das entregas, fazendo com que a otimização fique mais dinâmica e possívelmente acelerando o processo de entrega. O que também resulta em um aumento da satisfação dos clientes.

REFERENCES

[1] J. Allen, M. Piecyk, M. Piotrowska, F. McLeod, T. Cherrett, K. Ghali, T. Nguyen, T. Bektas, O. Bates, A. Friday, S. Wise, and M. Austwick. 2018. Understanding the impact of e-commerce on last-mile light goods vehicle activity in urban areas: The case of London. *Transportation Research Part D: Transport and Environment* 61 (2018), 325–338. https://doi.org/10.1016/j.trd.2017.07.020

[2] D-J Bertsimas and Garrett van Ryzin. 1991. A stochastic and dynamic vehicle routing problem in the Euclidean plane. *Operations Research* 39, 4 (1991), 601–615.

[3] John R. Birge and François Louveaux. 1997. *Introduction to Stochastic Programming*. Springer-Verlag, New York, NY, USA.

[4] C. Bălan. 2012. Net promoter score: Key metric of customer loyalty. *Quality - Access to Success* 13 (01 2012), 101–110.

[5] G Clarke and J-W Wright. 1964. Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research* 12, 4 (1964), 568–81. http://www.jstor.org/stable/167703

[6] J-F Cordeau, M Gendreau, G Laporte, J-Y Potvin, and F Semet. 2002. A guide to vehicle routing heuristics. *Journal of the Operational Research Society* 53, 5 (2002), 512–522. https://doi.org/10.1057/palgrave.jors.2601319

[7] Raphaëlle Ducret. 2014. Parcel deliveries and urban logistics: Changes and challenges in the courier express and parcel sector in Europe — The French case. *Research in Transportation Business & Management* 11 (2014), 15–22. https://doi.org/10.1016/j.rtbm.2014.06.009 Managing Freight in Urban Areas.

[8] Juan Camilo Fonseca-Galindo, Gabriela de Castro Surita, José Maia Neto, Cristiano Leite de Castro, and André Paim Lemos. 2020. A Multi-Agent System for Solving the Dynamic Capacitated Vehicle Routing Problem with Stochastic Customers using Trajectory Data Mining. arXiv:2009.12691 [cs.AI]

[9] Loggi. 2021. *loggiBUD: Loggi Benchmark for Urban Deliveries*. GitHub.

[10] A. Mikhalev and I.V. Oseledets. 2018. Rectangular maximum-volume submatrices and their applications. *Linear Algebra Appl.* 538 (Feb 2018), 187–211. https://doi.org/10.1016/j.laa.2017.10.014

[11] Oren E. Nahum and Yuval Hadas. 2009. Developing a model for the stochastic time-dependent vehicle-routing problem. In *2009 International Conference on Computers Industrial Engineering*. 118–123. https://doi.org/10.1109/ICCIE.2009.5223595

[12] Mohd Kamarul Irwan Abdul Rahim, Yiqing Zhong, El-Houssaine Aghezzaf, and Tarik Aouam. 2014. Modelling and solving the multiperiod inventory-routing problem with stochastic stationary demand rates. *International Journal of Production Research* 52, 14 (2014), 4351–4363. https://doi.org/10.1080/00207543.2014.883470 arXiv:https://doi.org/10.1080/00207543.2014.883470

[13] Stefan Ropke. 2005. *Heuristic and exact algorithms for vehicle routing problems*. Ph.D. Dissertation. Universidade de Copenhague, Copenhague, Dinamarca.

[14] Shabnam Sangwan. 2018. Literature Review on Travelling Salesman Problem. *International Journal of Research* 5 (06 2018), 1152.

[15] Yutong Zhang and Jing Liu. 2021. Emergency Logistics Scheduling Under Uncertain Transportation Time Using Online Optimization Methods. *IEEE Access* 9 (2021), 36995–37010. https://doi.org/10.1109/ACCESS.2021.3061454

Um Problema Integrado de Planejamento da Produção e da Distribuição de Produtos Manufaturados

Sasha K D A B H Barros
Programa de Pós-Graduação em Eficiência Energética e Sustentabilidade
Universidade Federal do Mato Grosso do Sul
Campo Grande MS Brasil
sasha.hohlenwerger@ufms.br

Willy A Oliveira Soler
Programa de Pós-Graduação em Eficiência Energética e Sustentabilidade
Universidade Federal do Mato Grosso do Sul
Campo Grande MS Brasil
willy.oliveira@ufms.br

RESUMO

O presente artigo tem por objetivo apresentar os resultados da pesquisa de mestrado em andamento na qual estuda-se um problema de planejamento conjunto da produção e da distribuição de produtos manufaturados. O problema abordado é baseado no sistema de produção adotado por algumas indústrias alimentícias brasileiras e cujas características também podem ser encontradas em indústrias de fundição, tecelagem de tapetes e panificação. Mais especificamente, considera-se um ambiente produtivo em que se observam as seguintes características: tempos e custos de preparação para produção; relevantes custos para manutenção de produtos em estoque; uma única linha de produção; distribuição dos itens produzidos realizada por uma frota homogênea de veículos. O problema abordado ainda se caracteriza pela flexibilidade no atendimento das demandas, ordens indivisíveis e janela de tempo de atendimento. Neste trabalho, apresentamos um modelo matemático que foi construído para representação do problema estudado. Por fim, apresentamos um cronograma contendo as etapas já realizadas e propostas futuras para a conclusão da pesquisa.

CONCEITOS CCS

• Computação aplicada • Pesquisa operacional • Análise de decisão • Otimização monocritério e tomada de decisão

PALAVRAS-CHAVES

Dimensionamento de lotes, Flexibilidade de atendimento das demandas, Problemas de roteamento de veículos

1 Introdução

Com o aumento da competitividade na indústria, percebeu-se a necessidade de reduzir os custos envolvidos nos processos, desde a produção até a entrega dos produtos ao cliente. Para tanto, tornou-se importante o desenvolvimento de ferramentas para um planejamento operacional eficiente, visando aumentar a eficiência da cadeia de suprimentos e tornar a empresa competitiva no mercado, além de reduzir o preço dos produtos finais. Esta busca por soluções mais eficientes levou a este estudo integrado do Problema de Dimensionamento de Lotes (PDL) [14] e do *Production Routing Problem* (PRP) [15].

Sendo assim, neste artigo tratamos de um problema de planejamento conjunto da produção e da distribuição de produtos, mais especificamente no sistema de produção adotado por indústrias alimentícias brasileiras com planta única de produção e capacidade limitada, que deve ser usada para produzir um vasto catálogo de produtos que são entregues aos clientes usando uma frota de veículos homogênea, sem atrasos ou entrega parcial. Essas características também podem ser encontradas em indústrias de carne embaladas, indústrias de panificação, indústrias de fundição e indústrias de tecelagem [14] e [15].

Esta pesquisa, considera características do PDL consideradas em [14]: i) Custos de manutenção de estoque; ii) Tempo e custo para configuração de *setup*; iii) Flexibilidade na escolha das demandas que serão atendidas: clientes fazem os pedidos antecipados com diversidade de itens. Devido às restrições da capacidade associada ou não ao tempo e custo de *setup*, pode ser menos lucrativo ou até mesmo inviável aceitar os pedidos dos clientes. As restrições de capacidade de produção podem ser impactadas pelo *lead time* depende do volume de demandas e, portanto, quanto mais pedidos são aceitos, os períodos de produção serão maiores, resultando na possibilidade de perda de prazos das entregas. E quando o alto custo de *setup*, não justifica a abertura da linha de produção, dado um pedido específico em que não é possível agregar demandas adicionais; iv) Pedidos indivisíveis e janelas de tempo de entrega: as demandas são dadas em forma de pedidos indivisíveis que, caso aceitos, devem ser completamente entregues, em um único período, dentro de um prazo estabelecido pelo cliente. Além disso, considera-se as características v) Planta única de produção com capacidade limitada; vi) Frota de veículos homogênea. Ambas as características também foram consideradas em [15].

Os modelos da literatura conseguem lidar com os problemas de armazenamento e o problema de distribuição, mas a flexibilidade

e a abordagem simultânea de todas essas características não são comumente abordadas na literatura. Para tanto, nesta pesquisa serão estendidos modelos matemáticos já propostos na literatura e desenvolvidos métodos heurísticos específicos que consideram a estrutura do problema abordado.

Este artigo tem por objetivo expor os resultados alcançados com a conclusão das etapas iniciais desta pesquisa e apresentar a perspectiva das próximas etapas deste trabalho e os resultados esperados.

2 Revisão da literatura

O PDL consiste em determinar um plano de produção em um determinado período de um horizonte de planejamento finito de forma que atenda as demandas dos clientes, minimizando os custos envolvidos no processo de produção [14].

A maioria dos modelos propostos na literatura para o PDL são extensões dos modelos clássicos *Discrete Lot Sizing and Scheduling Problem* (DLSP), *Continuous Setup Lot-Sizing Problem* (CSLP), *Problem Proportional Lot Sizing and Scheduling Problem* (PLSP), *Capacitated Lot-Sizing Problem* (CLSD) e *General Lot Sizing and Scheduling Problem* (GLSP) [7]. Por permitir a produção de diversos itens dentro de um mesmo período o CSLD é considerado um modelo geral que se vale de variáveis e restrições oriundas do Problema do Caixeiro Viajante (PCV).

A estrutura do CLSD tem sido utilizada para modelar problemas de diversos setores, como por exemplo em [2] em que foram propostas soluções para lidar com problemas de indústrias de recipientes de vidro, em [10] para indústria de iogurtes e [16] para indústria de semicondutores.

Produção e a distribuição são duas atividades críticas e interligadas, sendo um conjunto de ações que possibilitam a relação entre a indústria e o cliente, a busca pela otimização nestas atividades é chamado de Problema de Distribuição e Produção (PDP) [4]. O PDP pode ser adaptado nos três níveis de planejamento do gerenciamento de uma cadeia de suprimentos: estratégico, tático e operacional, sendo as decisões estratégicas vitais para a sobrevivência da empresa [11].

Desta forma o PDP consiste em tomadas de decisões cujo objetivo é minimizar os custos de produção e distribuição, através do planejamento da produção, atendimento de pedidos e fluxo de produção, de forma que atenda a demanda total do cliente respeitando a capacidade de produção da fábrica e de transporte dos veículos [15]. Sendo ele abordado diversas vezes na literatura como em [9] estudou um problema com vários produtos e uma planta única de produção capacitada. [1], [3], [5], e [6] abordaram apenas um único produto e uma frota com limite de rota por veículos em cada período. Problemas com múltiplas plantas de produção são abordadas em [8], [12] e [13]. Sendo que em [8] e [12] a frota do problema é heterogênea.

3 Modelo Matemático

Neste artigo, estendemos o modelo (CLSD-DTI) proposto em [14] e baseou-se no modelo (PRP) apresentado em [15] a fim de incorporar o roteamento de veículos no novo modelo formulado.

O CLSP-DTI-PRP consiste em determinar simultaneamente: i) as ordens dos clientes que serão aceitos, ii) o período (na respectiva janela de tempo) em que cada pedido aceito será entregue, iii) o veículo que será utilizado para a entrega de cada pedido aceito; iv) as rotas de entrega dos veículos utilizados. O objetivo é maximizar o lucro obtido pelos pedidos aceitos, descontando os custos incorridos pela manutenção do estoque, configuração de *setup*, da utilização dos veículos e o transporte das entregas. O modelo proposto (CLSD-DTI-PRP) consiste em maximizar a função objetivo (1) sujeita às restrições de (2) a (12).

Para modelar o problema, definimos sua representação, parâmetros e variáveis, consideramos $G:(F,E)$ um grafo completo, sendo $F:\{0,1,\dots,N\}$ e $E:\{(k,n)|k,n \in F, k \neq n\}$ onde o nó 0 representa a fábrica e os demais representam os clientes. Índices, parâmetros e variáveis são os seguintes:

Índices

J: Número de itens (indexados por $j \in \{1,\dots,J\}$);
T: Número de períodos (indexados por $t \in \{1,\dots,T\}$);
V: Número de veículos (indexados por $v \in \{1,\dots,V\}$);
N: Planta (indexados por $k = 0$), e número de pedidos (indexados por $n \in \{1,\dots,N\}$).

Parâmetros

F_n, L_n: Primeiro e último períodos da janela de tempo de entrega do pedido n;
P_{nt}: Lucro da ordem n no período $t \in \{F_n,\dots,L_n\}$;
O_t: Conjunto de pedidos que podem ser entregues no período t, ou seja, O_t: $\{n \mid F_n \leq t \leq L_n\}$;
q_{jn}: Quantidade do item j na ordem n;
C_t: Capacidade de produção no período t;
h_j: Custo de manutenção de estoque unitário do item j;
a_j: Tempo de processamento unitário do item j;
p_j: Peso unitário do item j;
st_j: Tempo para preparar o *setup* do item j;
sc_j: Custo para preparar o *setup* do item j;
vc_v: Capacidade do veículo v;
us_v: Custo de utilização do veículo v;
tc_{kn}^v: Custo transporte no limite da rota (k,n);

Variáveis

$x_{jt} \geq 0$: Quantidade do item j produzido no período t;
$I_{jt} \geq 0$: Estoque do item j no final do período t;
$\gamma_{nt} \in \{0,1\}$: 1 se o pedido n for entregue no período $t \in \{F_n,\dots,L_n\}$, e 0 caso contrário;
$z_{jt} \in \{0,1\}$: 1 se o *setup* do item j está configurado no período t; e 0 caso contrário;
$y_{nvt} \in \{0,1\}$: 1 se o pedido n é alocado no para o veículo v no período t; e 0 caso contrário;
$w_{knt}^v \in \{0,1\}$: 1 se o veículo v viajar através do limite da rota (k,n) no período t; e 0 caso contrário;
$G_{vnt} \geq 0$: Variáveis para representar ordem que veículo v atende o cliente n no período t.

O modelo CLSD-DTI-PRP é, então, dado por:

$$max \sum_{n=1}^{N} \sum_{t=F_n}^{L_n} P_{nt} \cdot \gamma_{nt} - \sum_{t=1}^{T} \sum_{j=1}^{J} h_j \cdot I_{jt} - \sum_{t=1}^{T} \sum_{j=1}^{J} sc_j \cdot z_{jt} - \sum_{v=1}^{V} \sum_{t=1}^{T} \sum_{n=1}^{N} us_v \cdot w_{0nt}^{v0} - \sum_{v=1}^{V} \sum_{t=1}^{T} \sum_{\substack{k=0,\\ k\neq n}}^{N} tc_{kn}^{v} \cdot w_{knt}^{v} \quad (1)$$

s.a.

$$I_{j,t-1} + x_{jt} = \sum_{n \in O_t} q_{jn} \cdot \gamma_{nt} + I_{jt}, \forall\, j, t \quad (2)$$

$$\sum_{t=F_n}^{L_n} \gamma_{nt} \leq 1, \forall\, n \quad (3)$$

$$\sum_{j=1}^{J} (a_j \cdot x_{jt} + st_t \cdot z_{jt}) \leq C_{t,}\ \forall\, t \quad (4)$$

$$x_{jt} \leq \frac{C_t}{a_j} z_{jt}, \forall\, j, t \quad (5)$$

$$\sum_{n=1}^{N} \sum_{j=1}^{J} (p_j \cdot q_{jn} \cdot y_{nvt}) \leq vc_{v,}\ \forall\, v, t \quad (6)$$

$$\sum_{v=1}^{V} y_{nvt} = \gamma_{nt}, \forall\, n, t \quad (7)$$

$$\sum_{\substack{k=0,\\ k\neq n}}^{N} w_{knt}^{v} = y_{nvt}, \forall\, v, t, n \quad (8)$$

$$\sum_{n=1}^{N} w_{0nt}^{v} \leq 1, \forall\, v, t \quad (9)$$

$$\sum_{n=1}^{N} w_{0nt}^{v+1} \leq \sum_{n=1}^{N} w_{0nt}^{v}, \forall\, v = 1, \ldots, V-1, t \quad (10)$$

$$\sum_{\substack{i=0,\\ i\neq n}}^{N} w_{int}^{v} = \sum_{\substack{m=0,\\ m\neq n}}^{N} w_{nmt}^{v}, \forall\, v, t, n \quad (11)$$

$$G_{vnt} \geq G_{vkt} + 1 - N(1 - w_{knt}^{v}), \forall\, v, k, n, t \quad (12)$$

A função objetivo (1) consiste em maximizar o lucro obtido pelos pedidos aceitos descontando a soma dos custos de manutenção e *setup* de estoque e os custos de distribuição. As restrições (2) são as restrições de balanceamento de estoque e as restrições (3) garantem que os pedidos sejam entregues no máximo uma vez dentro de suas respectivas janelas de tempo. As restrições (4) são restrições de capacidade da produção, enquanto as restrições (5) garantem que um produto só pode ser produzido se a linha de produção estiver configurada para tal. As restrições (6) são restrições de capacidade dos veículos, enquanto as restrições (7) estabelecem que cada pedido seja alocado integralmente em um único veículo. As restrições (8) garantem que todos os pedidos alocados em um veículo estejam na rota de entrega, enquanto as restrições (9) asseguram que um veículo visite um cliente no máximo uma vez dentro do período. As restrições (10) garantem um sequenciamento crescente ao escolher quais veículos irão atender determinada demanda no período, visto que a frota é homogênea ocorreria desperdício de recurso computacional ao realizar combinações aleatórias das ordens dos veículos a serem utilizados. As restrições (11) garantem o fluxo das rotas, assegurando a melhor rota de entregas e que encerre na planta e (12) são as restrições MTZ para eliminar *sub-tours*.

4 Resultados iniciais

O objetivo desta pesquisa é propor um modelo matemático e métodos de solução para um problema integrado de planejamento da produção e da distribuição de produtos. A fim de se alcançar o objetivo se elaborou as etapas:
1) Realizar uma ampla revisão bibliográfica sobre os métodos normalmente utilizados para solução de problemas de planejamento da produção e planejamento da distribuição; 2) Estender o modelo matemático proposto em [14] para considerar o problema de planejamento da distribuição; 3) Combinar as instâncias de teste propostas em [14] e [15] a fim de obter instâncias representativas para o problema estudado; 4) Desenvolver métodos heurísticos de solução baseados em busca local e na meta-heurística *variable neighbourhood search* (VNS) visando obter soluções factíveis de boa qualidade em aceitável tempo computacional; 5) Realizar testes computacionais com o objetivo de estudar a eficiência das abordagens propostas; 6) Realizar o exame de qualificação; e 7) Realizar a confecção e defesa do Trabalho de Conclusão de Curso (TCC).
A Etapa 1 foi atendida através do desenvolvimento de uma Revisão Sistemática da Literatura (RSL) que investigou quais são os impactos e soluções que a Pesquisa Operacional (PO) vêm causando na sociedade. Constatando que os aspectos ambientais e os econômicos são os mais promissores, onde as abordagens da PO, com a integração dos recursos computacionais e as Tecnologias da Informação e Comunicações (TICs) apresentam os resultados mais satisfatórios nas indústrias, sendo as questões sociais, da qual também compreende o desenvolvimento sustentável, as mais complexas de serem englobas, onde está sendo preparado um artigo para publicação de tais resultados. Além desta RSL, foi desenvolvido o modelo matemático CLSD-DTI-PRP, inédito na literatura, apresentado na Seção 3 deste artigo.

5 Próximas atividades

Como observado na seção 4, as duas primeiras etapas desta pesquisa estão concluídas. Agora, na terceira etapa, estamos trabalhando na geração de instâncias representativas para o problema estudado. A obtenção das instâncias está partindo do

dados de [14] e sendo analisando as possibilidades de combinar com as instâncias de [15] e/ou o *dataset* disponível no *benchmark* Loggi BUD.

O modelo encontra-se na fase de implantação computacional utilizando a ferramenta *Concert Technology* do *solver* comercial Cplex 20.1, na linguagem C++.

O andamento da pesquisa está representado no cronograma demostrado na Tabela 1 a seguir:

Etapas	2021							2022			
	J	J	A	S	O	N	D	J	F	M	A
1) Revisão	x	x			x	x					
2) Modelo		x	x	x	x						
3) Instâncias			x	x	x	x	x	x	x		
4) Heurísticas								x	x	x	x
5) Testes								x	x	x	x

Etapas	2022								2023		
	M	J	J	A	S	O	N	D	J	F	M
4) Heurísticas	x	x	x	x		x	x	x	x	x	
5) Testes	x	x	x	x		x	x	x	x	x	
6) Qualificação					x						
7) Confecção e defesa do TCC											x

Legenda A concluir Em andamento Concluído

Tabela 1: Cronograma da pesquisa

A etapa 4 em que serão desenvolvidos métodos heurísticos de solução, será iniciada em conjunto com a 5 em que serão realizados testes computacionais para comprovar a eficácia do modelo, ambas serão divididas em duas fases, a primeira se inicia próximo ao encerramento da etapa 3, que está em andamento e a segunda, logo após conclusão do estágio 6, quando ocorrerão orientações para ajustes e correções. Após o encerramento da segunda fase das etapas 4 e 5, será executada e etapa 7 que consistirá na confecção e defesa do TCC.

6 Conclusões e Perspectivas futuras

Neste artigo, apresentamos os resultados parciais de uma pesquisa de mestrado, em andamento, que está investigando um problema de planejamento integrado da produção e da distribuição que ocorre em diversos contextos industriais. Apresentamos um modelo matemático para o problema considerado, o qual está na fase de geração de dados e implementação computacional.

Nesta perspectiva, pretendemos dar continuidade ao estudo do problema descrito neste artigo desenvolvendo métodos heurísticos para obter soluções praticáveis, de qualidade e executáveis em tempo computacional admissível e em simultâneo realizar testes computacionais para verificar a eficiência das abordagens propostas.

Os resultados poderão beneficiar a solução de diversos problemas correlatos. Tais resultados serão descritos em forma de artigos que deverão ser submetidos para revistas científicas nacionais e internacionais.

AGRADECIMENTOS

A pesquisa tem o apoio da Universidade Federal de Mato Grosso do Sul e, em parte, do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq). A pesquisa está sendo financiada pela empresa Loggi.

REFERÊNCIAS

[1] Absi, Nabil, et al. "A two-phase iterative heuristic approach for the production routing problem." Transportation Science 49.4 (2015): 784-795.

[2] Almada-Lobo, Bernardo, et al. "Single machine multi-product capacitated lot sizing with sequence-dependent setups." *International Journal of Production Research* 45.20 (2007): 4873-4894.

[3] Archetti, Claudia, et al. "Analysis of the maximum level policy in a production-distribution system." *Computers & Operations Research* 38.12 (2011): 1731-1746.

[4] Bo, Vincenzo, et al. "Models and algorithms for integrated production and distribution problems." *Computers & Industrial Engineering* 154 (2021): 107003.

[5] Boudia, Mourad, Mohamed Aly Ould Louly, and Christian Prins. "Combined optimization of production and distribution." CD-ROM proceedings of the international conference on industrial engineering and systems management *IESM'05*. 2005.

[6] Boudia, Mourad, and Christian Prins. "A memetic algorithm with dynamic population management for an integrated production–distribution problem." *European Journal of Operational Research* 195.3 (2009): 703-715.

[7] Copil, Karina, et al. "Simultaneous lotsizing and scheduling problems: a classification and review of models." OR *spectrum* 39.1 (2017): 1-64.

[8] Darvish, Maryam, Homero Larrain, and Leandro C. Coelho. "A dynamic multi-plant lot-sizing and distribution problem." *International Journal of Production Research* 54.22 (2016): 6707-6717.

[9] Fumero, Francesca, and Carlo Vercellis. "Synchronized development of production, inventory, and distribution schedules." *Transportation science* 33.3 (1999): 330-340.

[10] Kopanos, Georgios M., Luis Puigjaner, and Christos T. Maravelias. "Production planning and scheduling of parallel continuous processes with product families." *Industrial & engineering chemistry research* 50.3 (2011): 1369-1378.

[11] Lee, Young Hae, and Sook Han Kim. "Production–distribution planning in supply chain considering capacity constraints." *Computers & industrial engineering* 43.1-2 (2002): 169-190.

[12] Lei, Lei, et al. "On the integrated production, inventory, and distribution routing problem." IIE *Transactions* 38.11 (2006): 955-970.

[13] Piewthongngam, Kullapapruk, Supachai Pathumnakul, and Suphakan Homkhampad. "An interactive approach to optimize production–distribution planning for an integrated feed swinecompany." *International Journal of Production Economics* 142.2 (2013): 290-301.

[14] Soler, Willy A.; Poldi, Kelly C.; Santos, Maristela O. Capacitated Lot Sizing and Scheduling with order acceptance and delivery time windows: Mathematical Model and a Mip-Based Heuristic. *Pesquisa Operacional*, v. 39, n. 3, p. 471-496, 2019.

[15] Watanabe, Henrique H.M., et al. Solution methods for the integrated production routing problem. *Revista de Sistemas de Informação da FSMA*, Vol. 19, p. 53 – 61, 2017.

[16] Xiao, Jing, et al. "A hybrid Lagrangian-simulated annealing-based heuristic for the parallel-machine capacitated lot-sizing and scheduling problem with sequence-dependent setup times." *Computers & Operations Research* 63 (2015): 72-82.

Roteirização no Território Inteligente

Matthieu Oliveira, Victor Ribeiro, Thais Batista, Everton Cavalcante, Marcel Oliveira
Universidade Federal do Rio Grande do Norte
Natal, Brasil
matthieuceo@gmail.com,cstavictor@gmail.com,thaisbatista@gmail.com,everton.cavalcante@ufrn.br,marcel@dimap.ufrn.br

RESUMO

Este trabalho faz parte de um projeto que visa propor uma abordagem multidimensional para análise do problema de roteirização considerando dados relacionados a diferentes domínios de uma cidade. No contexto do projeto, o foco desse trabalho implementar uma aplicação de roteirização usando uma plataforma georreferenciada para desenvolvimento de cidades inteligentes, a *Smart Geo Layers* - SGeoL. Especificamente, esse trabalho visa a criação da aplicação que contém uma camada com os dados da roteirização e que pode ser combinados com dados de quaisquer outras camadas do SGeoL que representem outros domínios de uma cidade. Dessa maneira, é possível concretizar uma análise multidimensional das melhores rotas para entrega de encomendas. A prova-de-conceito desse trabalho representa a camada de roteirização da cidade do Natal - RN.

PALAVRAS-CHAVE

logística, roteirização, território inteligente, plataforma georreferenciada, camadas de domínio

1 INTRODUÇÃO

Em uma cidade inteligente, vários sistemas gerenciam informações de diversos domínios, por exemplo, educação, saúde, segurança pública e transporte [7]. Cada sistema geralmente coleta e processa tais informações de forma independente dos demais, ainda que elas se refiram ao mesmo espaço geográfico. No entanto, um determinado domínio pode prover dados que podem também ser relevantes para outros domínios. A título de exemplo, a ocorrência de crimes em uma determinada região da cidade (ou seja, informações relacionadas ao contexto de segurança pública) pode ser determinante no traçado de rotas de entregas por transportadoras, que são informações do domínio logístico. Mesmo que tais sistemas sejam efetivos na coleta e gerenciamento dessas informações, a falta de integração entre eles acaba por impedir uma visão holística que poderia permitir correlacionar e analisar dados de forma mais sofisticada [1]. A análise vai além da tarefa de encontrar a rota mais rápida, uma vez que outras informações relevantes devem ser também consideradas, tais como ambiente, congestionamento e segurança do tráfego [8], de forma a garantir a eficácia, eficiência e segurança da operação de transporte.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
Workshop Virtual Loggi, 8 de dezembro de 2021, On-line
© 2021 Copyright held by the owner/author(s).

De forma a contribuir para resolver esse problema, o objetivo desse trabalho é especificar uma aplicação de roteirização que contém uma camada com dados de rotas para servir de suporte a um serviço de roteirização na cadeia logística. Tal serviço é objeto de estudo de outro projeto de pesquisa e desenvolvimento do mesmo grupo de autores desse artigo [5]. Essa camada está sendo elaborada sobre uma plataforma georreferenciada para o desenvolvimento de aplicações para cidades inteligentes, a plataforma *Smart Geo Layers - SGeoL* [6]. A SGeoL possibilita a realização de análises multidimensionais com dados de outras camadas existentes que representam domínios da cidade e que podem ter impacto na escolha de rotas para entrega de encomendas, a exemplo de dados sobre problemas de segurança e nas vias da cidade.

Esse trabalho está organizado da seguinte forma. A seção 2 comenta, brevemente, sobre trabalhos relacionados. A seção 3 apresenta a aplicação de roteirização. A seção 4 contém as conclusões e as atividades atuais.

2 TRABALHOS RELACIONADOS

Korczaka e Więcek [3] investigaram, através de uma revisão de literatura, trabalhos acerca de logística inteligente (*smart logistics*, em Inglês) sob a ótica do contexto de cidades inteligentes. Os autores ressaltam que dados obtidos de diferentes fontes devem ser detalhados, mensuráveis e coletados em tempo real e deve haver a possibilidade de agregá-los para uma análise conjunta. Esse trabalho confirma assim a importância do tratamento das questões de logísticas com informações de outros domínios de cidades inteligentes. O conceito de camada vai de encontro a essa necessidade, em que é possível sobrepor diferentes camadas, cada uma representando um tipo de informação, para que seja possível ter análises mais sofisticadas que consideram múltiplos domínios de uma cidade.

Kubek e Wiecek [4] apresentaram uma estratégia para tomada de decisão multicamadas de diversos problemas de logística, em especial para transporte. A estratégia proposta é a base para uma melhor integração e cooperação entre as entidades de logística da cidade, sendo centrada na integração de dados, inteligência ambiental e econômica, visando melhorar o transporte de cargas nas cidades e diminuindo o impacto ambiental. Nesse trabalho, problemas de logística são tratados sob a perspectiva de multicamadas visando especificamente o transporte com sustentabilidade ambiental. A proposta do presente trabalho apoia-se na mesma linha de integração de dados, porém com enfoque na constituição de uma camada relacionada à roteirização que pode ser correlacionada com outras camadas referentes a diversos domínios da cidade.

3 APLICAÇÃO DE ROTEIRIZAÇÃO

O objetivo da aplicação de roteirização proposta nesse trabalho é representar, em uma plataforma que integra dados de diversos domínios de uma cidade, a SGeoL, uma camada de roteirização que

Figura 1: Dashboard de Desenvolvimento mostrando camadas disponíveis na plataforma SGeoL.

mostre múltiplas rotas possíveis entre pontos. Tal camada pode ser relacionada com outras camadas que tenham informações de outros domínios da cidade de forma a ser determinada a(s) melhor(es) rota(s) entre dois pontos.

O SGeoL modulariza as informações de cada domínio de uma cidade usando o conceito *camada*. Cada camada é composta por um conjunto de elementos (entidades) que possuem atributos geo-espaciais. Através da sobreposição de diferentes camadas é possível correlacionar diferentes fontes de dados. Essa facilidade é especialmente interessante para a decisão de roteirização usando dados de uma cidade que podem impactar na rota, tais como dados sobre problemas de segurança (visando a integridade das encomendas), dados sobre problemas nas vias da cidade, entre outros.

A SGeoL provê um conjunto de funcionalidades para facilitar o desenvolvimento de aplicações para cidades inteligentes, integrando dados heterogêneos de diferentes domínios e posicionando-os sobre o território por meio de informações geográficas. Dentre os recursos da plataforma, o *Dashboard de Desenvolvimento*, ilustrado na Figura 2, fornece uma interface Web para auxiliar desenvolvedores na criação de novas aplicações, camadas e entidades de contexto, bem como no gerenciamento do controle de acesso às camadas e importação de dados. Por sua vez, o *Dashboard de Visualização* fornece aos usuários uma interface Web para visualizar entidades de contexto sobre o espaço urbano da cidade, além de possibilitar a correlação de informações de diferentes camadas. O formato de dados usado na SGeoL para representar camadas é o GeoJSON [2], um padrão aberto proposto como uma extensão do popular formato JSON, ao qual adiciona elementos que podem ser localizados no espaço geográfico, como pontos (incluindo endereços e localizações), linhas (incluindo ruas e estradas) e polígonos (para delimitar áreas no território), bem como suas propriedades.

Para a criação da camada de roteirização no SGeoL, esse trabalho usa o *Dashboard de Desenvolvimento* do SGeoL e importa, via requisições HTTP POST para a plataforma, as instâncias geradas a partir do Loggi-BUD e convertidas para o formato GeoJSON (detalhes desse processo podem ser encontrados em [5]). Tais instâncias são representadas como uma nova camada, *Routes* que pode ser exibida no *Dashboard de Visualização*. A Figura 3 mostra a camada *Routes* na SGeoL, contendo as rotas para a região metropolitana de Natal-RN.

4 CONSIDERAÇÕES FINAIS

Esse trabalho apresentou, brevemente, a criação de uma aplicação de roteirização em uma plataforma georreferenciada para aplicações de cidades inteligentes, a SGeoL. Especificamente, o trabalho relata a criação de uma camada de roteirização, que recebeu dados da cidade do Natal, e exibe possíveis múltiplas rotas entre pontos da cidade que são origem/destino de encomendas. Com essa camada disponibilizada no contexto do SGeoL, é possível realizar correlação entre dados de rotas com informações de outros domínios que tem impacto na escolha das melhores rotas.

A implementação desse trabalho está integrada com a de outro plano de trabalho, envolvendo os mesmos autores, que detalha todos os passos para geração de instâncias dos pontos de entrega para o Estado do Rio Grande do Norte e a Geração das rotas. Esses aspectos estão apresentados, detalhadamente, em [5].

Atualmente, o foco do presente trabalho está na melhoria da exibição das múltiplas rotas apresentadas na camada de roteirização e na continuidade da integração com os resultados do trabalho que envolve a abordagem multidimensional.

REFERÊNCIAS

[1] Everton Cavalcante, Nélio Cacho, and Thais Batista. 2017. Challenges to the development of smart city systems: A system-of-systems view. In *Proceedings of the 31st Brazilian Symposium on Software Engineering*. SBC, Fortaleza.

[2] Internet Engineering Task Force. 2016. RFC 7946 - The GeoJSON Format. https://tools.ietf.org/html/rfc7946.

[3] Jerzy Korczak and Kinga Kijewska. 2019. Smart Logistics in the development of smart cities. *Transportation Research Procedia* 39 (2019), 201–2011. https://doi.org/10.1016/j.trpro.2019.06.022

Figura 2: Dashboard de Visualização da plataforma SGeoL mostrando o Rio Grande do Norte.

Figura 3: Representação de rotas na camada *Routes* na plataforma SGeoL.

[4] Daniel Kubek and Paweł Więcek. 2019. An integrated multi-layer decision-making framework in the Physical Internet concept for the city logistics. *Transportation Research Procedia* 39 (2019), 221–230. https://doi.org/10.1016/j.trpro.2019.06.024

[5] Matthieu Oliveira, Victor Ribeiro, Thais Batista, Everton Cavalcante, and Marcel Oliveira. 2021. Uma Abordagem Multidimensional para Roteirização noTerritório Inteligente. In *Anais do Workshop da Loggi*. Loggi.

[6] Jorge Pereira, Thais Batista, Everton Cavalcante, Arthur Souza, Frederico Lopes, and Nélio Cacho. 2022. A platform for integrating heterogeneous data and developing smart city applications. *Future Generation Computer Systems* 128 (March 2022), 552–566. https://doi.org/10.1016/j.future.2021.10.030

[7] Eduardo Felipe Zambom Santana, Ana Paula Chaves, Marco Aurelio Gerosa, Fabio Kon, and Dejan S. Milojičić. 2017. Software platforms for smart cities: Concepts, requirements, challenges, and a unified reference architecture. *Comput. Surveys* 50, 6 (Nov. 2017). https://doi.org/10.1145/3124391

[8] Eiichi Taniguchi, Russell G. Thompson, Tadashi Yamada, and Ron van Duin. 2001. *City logistics: Network modeling and intelligent transport systems.* Emerald Publishing, United Kingdom.

Seleção de Rotas Modelos para resolver o Last-Mile Incremental Capacitated Vehicle Routing Problem (ICVRP)

Ascânio Sávio de Araujo Neves
asan2@ic.ufal.br
Instituto de Computação, Universidade Federal de Alagoas
Maceió, Alagoas, BRA

Ruan Heleno Correa da Silva
rhcs@ic.ufal.br
Instituto de Computação, Universidade Federal de Alagoas
Maceió, Alagoas, BRA

Bruno Costa e Silva Nogueira
bruno@ic.ufal.br
Instituto de Computação, Universidade Federal de Alagoas
Maceió, Alagoas, BRA

Rian Gabriel Santos Pinheiro
rian@ic.ufal.br
Instituto de Computação, Universidade Federal de Alagoas
Maceió, Alagoas, BRA

RESUMO

O objetivo desse projeto é resolver o *Last-Mile Incremental Capacitated Vehicle Routing Problem* (Last-Mile ICVRP) utilizando um conjunto de pools contendo rotas modelos e um algoritmo de identificação de quadrantes para alocar as entregas em unidades de carregamento. Ao alocar as entregas, é calculado a menor distância entre a entrega atual, as entregas que já foram alocadas e as entregas presentes na rota modelo escolhida. Nos resultados parciais obtidos, o algoritmo desenvolvido conseguiu bons resultados ao ser comparado com os resultados de outros algoritmos.

KEYWORDS

Clusterização, Last Mile ICVRP

ACM Reference Format:
Ascânio Sávio de Araujo Neves, Ruan Heleno Correa da Silva, Bruno Costa e Silva Nogueira, and Rian Gabriel Santos Pinheiro. 2021. Seleção de Rotas Modelos para resolver o Last-Mile Incremental Capacitated Vehicle Routing Problem (ICVRP). In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUÇÃO

O *Last-Mile Incremental Capacitated Vehicle Routing Problem* (Last-Mile ICVRP) é um caso particular do problema SD-CVRP (Stochastic and Dynamic Capacitated Vehicle Routing Problems) [6]. O problema é denominado Last-Mile por caracterizar as entregas entre o centro de expedição e o destinatário final. Ele pode ser descrito como se segue. Dado um conjunto (histórico) de clientes atendidos anteriormente $H = \{v_1, \ldots, v_\ell\}$, um conjunto $\mathcal{U}$ de unidades de carregamento e um conjunto com n novos clientes $D = \{v_{\ell+1}, \ldots, v_{\ell+n}\}$, o Last-Mile ICVRP tem como objetivo alocar cada uma das entregas do conjunto D a uma unidade de carregamento. Cada demanda é alocada, uma por vez, em uma única unidade de carregamento de acordo com a ordem de chegada pré-definida. As unidades de carregamento tem sua capacidade máxima que, quando atingida, precisa ser despachada para entrega. O problema busca achar uma alocação das entregas às unidades de carregamento, de maneira que a distância percorrida pelos veículos de entrega seja minimizada.

De forma mais específica, a demanda do i-ésimo cliente ($v_{\ell+i}$) deverá ser alocada em uma unidade de carregamento $U_j \in \mathcal{U}$, de acordo com: (i) a distância entre o local de entrega de $v_{\ell+i}$ e a rota atual associada a U_j — tal rota é formada pelos elementos do conjunto $\{v_{\ell+1}, \ldots, v_{\ell+i-1}\}$ que foram alocados em U_j —; e (ii) disponibilidade de espaço na unidade U_j, *i.e.*, o volume das entregas na rota U_j deve respeitar a capacidade Q do veículo. Vale ressaltar que, cada alocação é realizada de acordo com informações extraídas do histórico de clientes atendidos anteriormente. Estas informações são obtidas do conjunto H.

De maneira geral, a resolução dos problemas relacionados ao SD-CVRP pode ser dividida em duas etapas. A primeira etapa, chamada de *planejamento*, ocorre de maneira *offline* e utiliza dados históricos de entrega para computar um conjunto de decisões pré-definidas. A segunda etapa, chamada de *execução*, ocorre em tempo real e utiliza as ações pré-definidas na etapa anterior assim como as informações sobre os eventos em tempo real (ex.: chegada de uma nova demanda) para gerar as rotas finais. A metodologia mais comum utilizada na literatura consiste em definir um *pool* com várias soluções possíveis (etapa de *planejamento*) e usar algoritmos de seleção de rotas na etapa de *execução*.

2 TRABALHOS RELACIONADOS

Tradicionalmente, os VRP dinâmicos e estocásticos são formulados como processos de Markov [5, 8, 9] ou como programação estocástica [1]. Ambas as metodologias modelam os dados do problema como variáveis aleatórias que seguem uma distribuição previamente conhecida. O objetivo é otimizar uma medida de risco (como o valor esperado, a variância ou o valor em risco condicional de alguma função de custo), sujeito à satisfação de algumas restrições [3]. Note que ambas as abordagens não estão alinhadas com o problema encontrado na Loggi.

A partir de apresentações institucionais da empresa Loggi, é possível identificar duas estratégias recentes para resolver o Last-Mile ICVRP. A primeira estratégia foi apresentada em [2] e usa uma combinação de agentes autônomos com mineração de dados. Nesta abordagem, os agentes autônomos modelam as rotas a serem construídas. Além disso, técnicas de mineração de dados são usadas

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference'17, July 2017, Washington, DC, USA
© 2021 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

para extrair informações estocásticas sobre a distribuição de pacotes e assim otimizar a alocação de cada pacote em uma das rotas. Para determinar em qual rota r novo pacote p será alocado, a seguinte função é maximizada: $bet(p, r) = \alpha * bet_{DataMining}(p, r) + (1 - \alpha) * bet_{Distance}(p, r)$, onde $bet_{DataMining}(p, r)$ é um valor extraído da mineração de dados, $bet_{Distance}(p, r)$ é uma função que mede a distância entre o pacote p e os pacotes já inseridos na rota r, e α é um valor no intervalo $[0, 1]$. Os resultados mostram que o número de rotas que precisam ser geradas pela abordagem proposta é em média superior em 17% quando comparado com o número de rotas gerado por um solver para a versão estática do *CVRP*.

A segunda estratégia da empresa para resolver o Last-Mile IC-VRP foi apresentada em video [7]. A partir do vídeo, é possível depreender que a estratégia consiste em duas etapas: uma etapa de clusterização rotas boas geradas a partir do histórico de entregas (*offline*), seguida de uma etapa alocação dos pacotes em um dos clusters de rotas (*online*). Três métodos de clusterização (*bag of cells*, distância euclidiana e distância entre ruas) e dois métodos de alocação de rotas (guloso e clusters artificiais) foram experimentados. A combinação do método de clusterização baseado em distância entre ruas com o de alocação usando clusters artificiais apresentou os melhores resultados. Estes resultados mostram que a solução apresentada é apenas 16% pior que as encontradas por meio de um solver para a versão estática do *CVRP*.

3 MÉTODO PROPOSTO

Para explicar o funcionamento do algoritmo desenvolvido, as principais funcionalidades foram divididas em sete tópicos.

Figura 1: Fluxograma do método proposto.

A Figura 1 mostra o fluxograma do algoritmo desenvolvido para resolver o *Last-Mile Incremental Capacitated Vehicle Routing Problem(ICVRP)*. Em vermelho, estão relacionados os processos referente à primeira etapa, chamada de *planejamento*, que ocorre de maneira *offline*. Em azul, estão relacionados os processos referente à segunda etapa, chamada de *execução*, que ocorre em tempo real.

3.1 Modelo de Clusterização

No algoritmo desenvolvido, é executado o algoritmo *K-Means* sobre as entregas com o valor pré-definido de k. O modelo é treinado a partir dos pontos das entregas presentes no histórico.

3.2 Geração do conjunto de Pools

Cada pool é composto pelas rotas modelos das entregas de um cluster do modelo de clusterização inicialmente criado com k clusters. Cada rota modelo é uma rota encontrada a partir do algoritmo que busca solucionar o CVRP.

Com isso, é feito a predição para identificar o cluster de cada entrega e assim separar as entregas do histórico no array do cluster identificado. Após a separação, cada array é repartido em lotes com uma quantidade de entregas pré-definida B. Para cada repartição, o algoritmo referente ao *CVRP* é chamado para buscar soluções nas entregas presentes daquela repartição. Com isso, as rotas modelos encontradas são armazenadas nos pools referente ao seu cluster.

3.3 Filtragem das Rotas Modelos

O algoritmo de filtragem das rotas modelos busca remover as rotas modelos que contêm entregas em quadrantes poucos frequentes.

No algoritmo desenvolvido, utilizamos a biblioteca *s2phere* que contém uma parte da biblioteca *geometry S2* implementada em *Python* para o mapeamento das células s2, com nível 12. Somamos a quantidade de entregas que estão situadas em cada quadrante e, de forma percentual e decrescente, ordenamos tais quadrantes. Depois disso, buscamos definir um percentual de corte, isto é, um valor que se o percentual do quadrante for menor o mesmo será considerado um quadrante pouco frequente. Com isso, há análise dos quadrantes em que cada rota modelo passa, caso identifique um quadrante considerado pouco frequente, a rota modelo será adicionada na lista de rotas a serem removidas daquele conjunto de pools. Por fim, antes de haver a remoção, analisamos se a quantidade de rotas modelos a serem removidas do pool é igual a quantidade de rotas presente naquele pool, caso seja igual, o valor do percentual de corte é diminuído e o processo de análise de quadrantes para selecionar as rotas modelos é reiniciado, caso não seja igual, as rotas modelos selecionadas são removidas.

3.4 Distribuição das Unidades de Carregamento

A distribuição das Unidades de Carregamento tem como objetivo fornecer a quantidade de unidades de carregamentos para cada cluster. Ao utilizar as entregas do histórico e separar tais entregas em seus determinados clusters, a distribuição é feita de forma proporcional à quantidade de entregas presentes nesses clusters em relação a quantidade total de entregas. Com isso, multiplicamos cada percentual pelo número de unidades de carregamento. Por ser um número inteiro é necessário depois fazer o balanceamento para que a quantidade de unidades de carregamento distribuídas seja igual ao número de unidades de carregamento inicial, além de buscar garantir que cada cluster tenha, no mínimo, uma unidade de carregamento. O balanceamento é feito removendo uma unidade de carregamento, por iteração, do primeiro cluster que tem mais de uma unidade de carregamento.

3.5 Armazenamento de Centróides

O algoritmo de armazenamento de centróides busca armazenar centróides a partir de modelos de clusterização tendo como objetivo designar cada centróide em uma unidade de carregamento. A partir dela, poderemos tratar as unidades de carregamento de forma diferente caso as mesmas ainda não tenham entregas alocadas. A partir do modelo com k clusters, é criado k modelos de clusterização, onde cada modelo terá a quantidade de clusters definida pelo algoritmo de distribuição de unidades de carregamento. Cada modelo com M_i

clusters terá M_i centróides. Com isso, armazenamos cada centróide em sua determinada unidade de carregamento.

3.6 Alocação das Entregas nas Unidades de Carregamento

O processo de alocação das entregas é feito a partir da análise das entregas em relação às rotas modelos e aos quadrantes presentes, e é organizado em três etapas:

(1) Análise do ponto da entrega atual em relação à rota modelo atual de cada unidade de carregamento.
(2) Análise do quadrante que o ponto da entrega atual está presente em relação aos pontos das outras entregas já alocadas.
(3) Cálculo do centróide ou cálculo do ponto da entrega atual em relação aos pontos da rota modelo escolhida e das entregas já alocadas em cada unidade de carregamento.

No Algoritmo 1, é mostrado o pseudocódigo referente ao cálculo da menor distância entre uma entrega e um conjunto de entregas.

Algorithm 1 Pseudocódigo referente ao cálculo da distância entre uma entrega e um conjunto de entregas

```
Require: delivery : Delivery, deliveries : [Delivery]
Ensure: dist_min ← inf
  for d in deliveries do
      dist ← distance(delivery.point, d.point)
      if dist < dist_min then
          dist_min ← dist
      end if
  end for
```

3.6.1 Análise das rotas modelos atuais: É verificado se o ponto da entrega atual está presente na rota modelo atual de alguma unidade de carregamento. Caso esteja, já adicionamos a entrega naquela unidade de carregamento.

No Algoritmo 2, é mostrado o pseudocódigo referente à análise das rotas modelos atuais. A entrada consiste da entrega atual, um array referente as unidades de carregamento do cluster identificado a partir da predição do ponto da entrega no modelo de clusterização.

Algorithm 2 Pseudocódigo referente ao análise das rotas modelos atuais

```
Require: delivery : Delivery, UCS_c : [UC]
Ensure: phi ← False, dist_min ← inf
  for UC in UCS_c do
      if delivery in UC.model_route then
          uc ← UC
          phi ← True
          model_route ← UC.model_route
          break
      end if
  end for
```

3.6.2 Análise dos quadrantes: É verificado se a entrega atual está no mesmo quadrante de alguma entrega já alocada. Caso esteja, alocamos na unidade de carregamento de menor distância. Esse algoritmo só será executado caso não esteja presente em nenhuma rota modelo atual.

No Algoritmo 3, é mostrado o pseudocódigo referente à análise dos quadrantes presentes nas entregas já alocadas e o quadrante da entrega atual. A entrada consiste da entrega atual, um array referente as unidades de carregamento do cluster, identificado a partir da predição do ponto da entrega no modelo de clusterização, e das células S2. A função *equalQuadrant* busca identificar a presença de quadrantes em comum entre o ponto da entrega atual e os pontos das entregas já alocadas daquela unidade de carregamento.

Algorithm 3 Pseudocódigo referente à análise dos quadrantes

```
Require: delivery : Delivery, UCS_c : [UC], Cells : [S2Cells]
Ensure: s2cells ← False, dist_min ← inf
  for UC in UCS_c do
      for d_uc in UC.deliveries do
          if equalQuadrant(delivery, d_uc) then
              dist ← distance(delivery.point, d_uc.point)
              if dist < dist_min then
                  dist_min ← dist
                  uc ← UC
                  s2cells ← True
                  model_route ← UC.model_route
              end if
          end if
      end for
  end for
```

3.6.3 Cálculo do centróide e da rota modelo: No Algoritmo 4, é mostrado o pseudocódigo referente ao cálculo do centróide e da rota modelo. A entrada consiste da entrega atual, um array referente as unidades de carregamento do cluster, identificado a partir da predição do ponto da entrega no modelo de clusterização, e do Pool referente ao cluster.

Caso o ponto da entrega atual não esteja nem nas rotas modelos atuais e nem no mesmo quadrante de alguma entrega já alocada, então a análise e escolha da unidade de carregamento é feita de duas maneiras:

(1) Se a unidade de carregamento estiver vazia, calculamos a distância da entrega atual para o centróide dela que definimos antes da alocação.
(2) Se houver entregas na unidade de carregamento, escolhemos a rota modelo, a partir do somatório das distâncias mínimas entre as entregas alocadas e as entregas das rotas modelos, e depois fazemos uma cópia das entregas junto das entregas da unidade de carregamento. Por fim, escolhemos a unidade de carregamento com a menor distância encontrada entre a entrega atual e as entregas da cópia feita.

A partir dos Algoritmos 1, 2, 3 e 4, podemos organizar o pseudocódigo do algoritmo de alocação das entregas nas unidades de carregamento, mostrado no Algoritmo 5. A entrada consiste de um array com as entregas a serem alocadas, um array referente

Algorithm 4 Pseudocódigo referente ao cálculo do centróide e da rota modelo:

```
Require: delivery : Delivery, UCS_c : [UC], pool : Pool
Ensure: dist_min ← inf
  for UC in UCS_c do
      if UC.size = 0 then
          dist ← distance(delivery.point, UC.centroide.point)
      end if
      if UC.size > 0 then
          model_route_esc ← modelRoute(pool, UC, delivery)
          copy_deliveries ← model_route.deliveries
          copy_deliveries.push(UC.deliveries)
          dist ← distDelivery(delivery, d_copy.deliveries)
      end if
      if dist < dist_min then
          dist_min ← dist
          uc ← UC
          model_route ← model_route_esc
      end if
  end for
```

à distribuição das unidades de carregamento, um array referente as unidades de carregamento, o modelo K-Means, um array referente aos pools, contendo as rotas modelos, e as células S2, para identificar os quadrantes.

Algorithm 5 Pseudocódigo referente à alocação das entregas nas unidades de carregamento

```
Require: deliveries : [Delivery], D : [int], UCS : [UC], Model :
  KMeans, P : [Pool], Cells : [S2Cells]
Ensure: routes ← []
  for delivery in deliveries do
      UCS_c ← []
      cluster ← Model.predict(delivery.point)
      for i in range(0,len(D)) do
          if D[i] = cluster then
              UCS_c.push(UCS[i])
          end if
      end for
      phi, uc, model_route ← ARMA(delivery, UCS_c)
      if phi not True then
          s2cells, uc, model_route ← AQ(delivery, UCS_c, Cells)
      end if
      if phi not True and s2cells not True then
          uc, model_route ← CRM(delivery, UCS_c, P[cluster])
      end if
      if (UCS[uc].size + delivery.size) > capacity_vehicles then
          routes.push(UCS[uc].deliveries)
          UCS[uc].pop(UCS[uc].deliveries)
      end if
      UCS[uc].push(delivery)
      UCS[uc].model_route ← model_route
  end for
```

3.7 Organização das rotas

O despacho de uma unidade de carregamento acontece quando a capacidade máxima da mesma é atingido. Com isso, é executado o algoritmo que busca uma solução para o *TSP(Traveling Salesman Problem)* em cada conjunto de entregas. No fim, o mesmo processo de despacho acontece nas unidades de carregamento que não ultrapassaram a capacidade, executando o determinado algoritmo para organizar as rotas.

4 RESULTADOS PARCIAIS/PRELIMINARES

O algoritmo do projeto foi desenvolvido na linguagem de programação *Python*. Foi utilizado o método *K-Means*, da biblioteca *Sklearn*, como método de clusterização, além da utilização da biblioteca *Numpy*. Houve a instalação do servidor *OSRM* para utilizar as distâncias do *OpenStreetMap* e a aplicação do algoritmo implementado com a ferramenta *OR-Tools*, do *Google*, que busca resolver o *Traveling Salesman Problem (TSP)* e *Capacitated Vehicle Routing Problems (CVRP)* disponibilizado pelo repositório *loggibud* [4], da *Loggi*, presente na plataforma *GitHub*.

Como forma de avaliação do algoritmo desenvolvido foi utilizado as instâncias cvrp, */train* e */dev*, dos diretórios *df-0*, *pa-0* e *rj-0* presentes no banco de dados disponibilizado pela *Loggi*. A pasta */train*, contém os dados que utilizamos como o histórico, distribuídos em 90 instâncias. A pasta */dev* é utilizada para gerar os resultados, de modo que ela contém 30 instâncias de testes.

O seguinte formato foi usado para encontrar um resultado a ser considerado geral de cada diretório:

(1) Inicialmente, separamos as entregas das instâncias de treino em batches (ou lotes) para que o algoritmo referente ao CVRP consiga encontrar uma solução. O tamanho do batch escolhido para os resultados foi 300, pois foi o tamanho que o algoritmo do arquivo */loggibud/v1/baselines/shared/ortools.py*, do repositório *loggibud*, consegue resolver em tempos razoáveis.
(2) Buscamos a distância percorrida das soluções encontradas para cada instância. As distâncias percorridas encontradas denominamos em resultados *off-line*.
(3) Executamos os algoritmos, de modo que a pegamos a distância percorrida das soluções geradas, utilizando a função *evaluate_solution*, presente no repositório *loggibud*, no arquivo */loggibud/v1/eval/task1.py*, e fazemos o cálculo da diferença em relação ao resultado obtido de cada instância do passo 1.
(4) Fazemos o cálculo da média dos percentuais, somando o cálculo da diferença encontrada entre o determinado algoritmo e o algoritmo que busca resolver o CVRP de cada instância (off-line) e dividimos pela quantidade de instâncias presente no diretório. O resultado do cálculo da média será considerado o resultado geral do diretório.

Como forma de comparação, utilizamos os resultados encontrados após a execução de dois algoritmos presentes no repositório *loggibud*, na pasta */loggibud/v1/baselines/task2*: *kmeans_greedy.py* e *qrp_sweep.py* com os valores de k no intervalo de 4 a 10.

4.1 Diretório Pará (pa-0)

O diretório pa-0 contém mais de 25000 entregas em seu histórico. Por ter uma média próxima de 300 entregas por instância, consideramos esse diretório como um diretório com pequenas instâncias, em comparação com os outros dois diretórios utilizados. Para os testes, foram 14 unidades de carregamento no algoritmo referente ao algoritmo desenvolvido.

Tabela 1: Resultados encontrados do diretório pa-0.

N_clusters	Algoritmo	KMeans_greedy	QRP_sweep
4	**17.25%**	**16.34%**	41.30%
5	18.62%	17.22%	24.61%
6	24.75%	23.89%	33.68%
7	25.47%	23.17%	34.86%
8	24.02%	22.44%	30.91%
9	47.92%	25.34%	25.99%
10	38.85%	26.72%	**20.79%**

A Tabela 1 apresenta os resultados referentes a instância pa-0. Nesta tabela, as colunas 2, 3, 4. representam os resultados, para cada quantidade de cluster, dos métodos: *algoritmo desenvolvido*, *kmeans_greedy.py* e *qrp_sweep.py*. Em negrito, destacamos os melhores resultados de cada método.

Figura 2: Resultados encontrados do diretório pa-0.

A Figura 2 busca mostrar o comportamento dos resultados encontrados de pa-0 dos três métodos no intervalo de número de clusters utilizado.

4.2 Diretório Distrito Federal (df-0)

O diretório df-0 contém mais de 88000 entregas em seu histórico. Por ter uma média próxima de 1000 entregas por instância, consideramos esse diretório como um diretório com instâncias médias, em comparação com os outros dois diretórios utilizados. Para os testes, foram 28 unidades de carregamento no algoritmo referente ao algoritmo desenvolvido.

De forma similar à Tabela 1 e à Figura 2, temos a Tabela 2 e a Figura 3, respectivamente. Eles apresentam os resultados os resultados encontrados e seu comportamento no intervalo de 4 a 10 clusters.

Tabela 2: Resultados encontrados do diretório df-0

clusters	Algoritmo	KMeans_greedy	QRP_sweep
4	16.97%	18.21%	25.56%
5	17.22%	17.22%	20.53%
6	17.74%	**12.04%**	23.30%
7	**16.27%**	15.41%	19.92%
8	19.42%	16.88%	18.19%
9	19.17%	16.92%	**18.09%**
10	19.65%	17.68%	21.56%

Figura 3: Resultados encontrados do diretório df-0.

4.3 Diretório Rio de Janeiro (rj-0)

O diretório rj-0 contém mais de 320000 entregas em seu histórico. Por ter uma média próxima de 3500 entregas por instância, consideramos esse diretório como um diretório com instâncias grandes, em comparação com os outros dois diretórios utilizados. Para os testes, foram 28 unidades de carregamento no algoritmo referente ao algoritmo desenvolvido.

Tabela 3: Resultados encontrados do diretório rj-0.

N_clusters	Algoritmo	KMeans_greedy	QRP_sweep
4	18.28%	25.67%	23.56%
5	18.78%	25.17%	23.87%
6	18.32%	24.93%	21.46%
7	18.60%	23.87%	21.44%
8	**18.17%**	**23.00%**	20.47%
9	18.99%	23.86%	**19.31%**
10	20.00%	23.56%	19.80%

Figura 4: Resultados encontrados do diretório rj-0.

De forma similar à Tabela 1 e à Figura 2, temos a Tabela 3 e a Figura 4, respectivamente. Eles apresentam os resultados os resultados encontrados e seu comportamento no intervalo de 4 a 10 clusters.

5 CONCLUSÃO

Neste projeto trouxemos uma alternativa de seleção de rotas modelos, com uso da identificação de quadrantes iguais, com o objetivo de ser uma alternativa para resolver o *Last-Mile Incremental Capacitated Vehicle Routing (Last-Mile ICVRP)*. Os resultados parcialmente obtidos mostraram que ao ser comparado com os outros dois algoritmos executados, o algoritmo desenvolvido teve bons resultados nos diretórios que contém instâncias consideradas pequenas ou médias, $pa - 0$ e $df - 0$, e teve melhores resultados no diretório que contém instâncias grandes, $rj - 0$.

6 TRABALHOS EM ANDAMENTO E FUTUROS

Na análise do desenvolvimento do algoritmo, entre os possíveis próximos trabalhos, pode-se relacionar:

(1) Nova forma de gerar o conjunto de pools contendo as rotas modelos.
(2) Desenvolvimento de um novo método de filtrar as melhores rotas modelos.
(3) Desenvolvimento de um novo método de distribuição das unidades de carregamento.
(4) Desenvolvimento de novo método de tratamento das rotas modelos.
(5) Desenvolvimento de novo método de tratamento de unidades de carregamento sem entregas alocadas.

Em relação aos trabalhos em andamento, está sendo estudado uma maneira de melhorar a escolha das rotas modelos no processo de alocação das unidades de carregamento.

REFERÊNCIAS

[1] Dimitris Bertsimas, Philippe Chervi, and Michael Peterson. 1995. Computational approaches to stochastic vehicle routing problems. *Transportation science* 29, 4 (1995), 342–352.
[2] Juan Camilo Fonseca-Galindo, Gabriela de Castro Surita, José Maia Neto, Cristiano Leite de Castro, and André Paim Lemos. 2020. A Multi-Agent System for Solving the Dynamic Capacitated Vehicle Routing Problem with Stochastic Customers using Trajectory Data Mining. *arXiv preprint arXiv:2009.12691* (2020).
[3] Chrysanthos E Gounaris, Panagiotis P Repoussis, Christos D Tarantilis, Wolfram Wiesemann, and Christodoulos A Floudas. 2016. An adaptive memory programming framework for the robust capacitated vehicle routing problem. *Transportation Science* 50, 4 (2016), 1239–1260.
[4] Loggi. 2021. loggiBUD: Loggi Benchmark for Urban Deliveries. https://github.com/loggi/loggibud.
[5] Warren B Powell. 1988. A Comparative review of alternative algorithms for the dynamic vehicle allocation problem. In *Vehicle routing: methods and studies. (Studies in Management Science and Systems, Volume 16)*, B. Golden and A. Assad (Eds.). Amsterdam ; New York : North-Holland, 249–291.
[6] Ulrike Ritzinger, Jakob Puchinger, and Richard F Hartl. 2016. A survey on dynamic and stochastic vehicle routing problems. *International Journal of Production Research* 54, 1 (2016), 215–231.
[7] Gabriela Surita and José C. Maldonado. 2020. Evolução das Soluções e Experiência de Roteirização na Loggi. https://www.youtube.com/watch?v=IDmaN7ima7k. [Online; acessado em 28 Março 2021].
[8] Barrett W Thomas and Chelsea C White Iii. 2004. Anticipatory route selection. *Transportation Science* 38, 4 (2004), 473–487.
[9] Marlin W Ulmer, Justin C Goodson, Dirk C Mattfeld, and Marco Hennig. 2019. Offline–online approximate dynamic programming for dynamic vehicle routing with stochastic requests. *Transportation Science* 53, 1 (2019), 185–202.

Resilient Routing

Arthur Souza Ribeiro
Universidade Federal de São Paulo (UNIFESP),
São José dos Campos, São Paulo, Brazil
arthur.souza@unifesp.br

Leonardo Bacelar Lima Santos
Universidade Federal de São Paulo (UNIFESP) & Centro Nacional de Monitoramento e Alertas de Desastres Naturais (Cemaden),
São José dos Campos, São Paulo, Brazil
santoslbl@gmail.com

Lívia Rodrigues Tomás
Centro Nacional de Monitoramento e Alertas de Desastres Naturais (Cemaden),
São José dos Campos, São Paulo, Brazil
liviatomas@gmail.com

ABSTRACT

Several delivery services have increased routes due to recurrent flooding in São Paulo city. This work uses new methods to visualize and predict how much a flood occurrence affects delivery routes. Furthermore, it verifies whether it is a critical node in network mobility, considering the complexity of the street network and the graph theory. The approaches consist of implementing the vulnerability index, which is already known in graph theory, and creating a new index specifically for that purpose called Shortest Paths Without a Node (SPAWN). These indexes can measure how much the network mobility is impacted, and the increase of the route's length flooding occurs.

KEYWORDS
Routing, Floods, Graph theory, Complex Networks, Vulnerability

ACM Reference format:

Arthur Souza Ribeiro, Leonardo Bacelar Lima Santos e Lívia Rodrigues Tomás. 2021. Resilient Routing. *São José dos Campos, SP, Brazil, 5 pages.*

1 INTRODUCTION

Extreme events around the world have increased significantly in recent years. Among all-natural disasters, floods are the most frequent. The number of people affected by floods and financial-economic and insured damages has increased too. Affected people include those who live and those who pass through the flooded area [1,2].

The interruption of vehicular traffic and the flow of people causes changes in the routes used, sometimes increasing travel time or even preventing people from reaching their final destination, such as the workplace or their homes. Thus, the quantification of the impact of exceptions on route length is essential for urban planning.

Our main contribution is to propose indicators of the potential impact of these events.

The study area is in downtown São Paulo, and it has a rain gauge that measures the amount and intensity of rainfall that can trigger floods and flash floods. The complexity of the road network of this area is a good case study as it represents the challenges faced by the Loggi company.

2 OBJECTIVES

The main goal of this paper is to evaluate the potential impact of exceptions, such as heavy rain and urban flooding, on the length of urban routes.

2.1 Specific objectives

There are four specific goals on this work:

1. Analyze a set of occurrences of flooding in 2019 in São Paulo city, Brazil.
2. Promote the geographic visualization of data, from flood occurrence to street networks.
3. Develop computational codes using graph theory for route analysis.
4. Propose a new index that quantifies the impact of exceptions on the length of urban routes.

3 DEVELOPED ACTIVITIES

In the beginning, our work team had scheduled all the intended phases of the project for each month. The entire project lasted nine months. Because of the COVID-19 pandemic, all the work was made by home, which did not have affected the schedule.

Table 1 shows the developed activities from month 1 to month 9.

Table 1 – Developed activities

Month 1	Alignment of the project with possible comments from the Loggi company, and studies about the environmental and climate data.
Month 2	Use of real data on the occurrence of flooding in São Paulo city.
Month 3	Construction and analysis of the street graph.

Month 4	Formulation of the routing model incorporating environmental variables and climatic.
Month 5	Solution of the routing model, considering a computationally viable instance for initial tests.
Month 6	Discussion of preliminary results with representatives of the Loggi company and possible adjustments to the model.
Month 7	Refined routing model solution and considering the widest possible instance.
Month 8	Review of all project documentation, evaluation of the possibility of including dynamic and stochastic components.
Month 9	Preparation of final report and final presentation to representatives of the Loggi company.

4 IMPLEMENTATION

4.1 The traditional and the new index

There are several papers in the graph theory literature on street networks [3,4]. Among the indexes on this area, there is the vulnerability [5]. The vulnerability index is related to the impact of a node removal on the topological efficiency of the network [6]. When a node is taken off, the path length between any pair of nodes in the network will never be shorter than in the original situation.

Mathematically, the efficiency of a pair of nodes is:

$$E_{ij} = \frac{1}{d_{ij}} \qquad (1)$$

, where d_{ij} is the shortest path length between nodes i and j. In a graph G with N nodes, the global efficiency is defined by:

$$E(G) = \frac{1}{N(N-1)} \sum_{i \neq j \in G} \frac{1}{d_{ij}} \qquad (2)$$

The vulnerability index of a node is:

$$V(k) = \frac{E(G) - E(G \backslash k)}{E(G)} \qquad (3)$$

, where $E(G \backslash k)$ is the global efficiency of the graph G, removing the node k.

In this paper, we propose a new index: the SPAWN (Shortest Paths Without a Node) index, that measures how long is the set of all paths in a graph when a node is taken off.

Mathematically:

$$M(G,k) = \sum_{\{\forall\, i,\ \ j \in G\,;\ \ i,\ \ j \neq k\}} d_{ij} \qquad (4)$$

, where $M(G,k)$ is the sum of all values in a matrix that has all the shortest paths lengths of a graph G, disregarding the line and column k, and d_{ij} is the shortest possible distance between nodes i and j.

The SPAWN index associated to a node k, $S(k)$, is:

$$S(k) = M(G \backslash k, k) - M(G,k) \qquad (5)$$

, where $M(G \backslash k, k)$ is the sum of the new shortest paths with node k removed and $M(G,k)$ is the sum of the shortest paths ignoring node k.

4.2 Computational Experiment

Using the python library OSMNX [7], we can manipulate the shapefile of street networks in the study area (center of the city of São Paulo, Brazil). The graph used in the case study has 1059 nodes and 1838 edges.

Using the traditional Dijkstra's algorithm, finding the shortest path between any pair of nodes is possible. When a node in the path is taken off, it is possible to search again - considering the new graph: the original one with all nodes but the removed one. Figure 1 shows the main idea behind the index SPAWN. The path in red is the original one between a selected pair of nodes, and the path in yellow is considering the removal of a flooded street in the original path between the same pair of nodes. Obviously, the second path is equal or, often, longer than the first.

5 PARTIAL RESULTS AND DISCUSSION

Both parameters were calculated with a graph of 1.4km radius in São Paulo city downtown. The vulnerability index is shown in Figure 2. The greater the node's vulnerability, the greater the impact on the network. In our case, when we look at a node with a high vulnerability, we see that its value is close to 0.025, and if we multiply it by 100, we see that its impact is 2.5% on network mobility.

Although this is a relatively low number, we should also look at how much the total lengths of routes increase without this node. For this, we use the SPAWN index that can be graphically visualized in Figure 3. The SPAWN index reveals the total increase in the lengths of new routes without a particular node. For didactic purposes, we compared the same previously mentioned node of vulnerability 0.025, but now also looking at its SPAWN parameter. The same node, with 2.5% vulnerability has a total increase in routes worth 75,000 kilometers, representing a considerable impact on specific routes.

We have to say that this happens for some cases of relevant nodes in the network. However, for the most part, the vulnerability and SPAWN of the nodes do not have high values, just looking at Figures 2 and 3.

We also generated the visualization of the 2019 flood points within a radius of 1.4 km, as shown in Figure 4. Looking at the three images, we can see that some flood points have high vulnerabilities and SPAWN.

6 WORKING IN PROGRESS AND FUTURES

Currently, we are working on reducing the computational cost of the proposed index. In the following steps, we are interested in conducting a statistical test to analyze if the most vulnerable streets (using the vulnerability and the SPAWN indexes) are also where floods are often - the combination of the two features brings very complicated risk scenarios.

Figure 1: Red route without removing the flood points and the yellow route with removing the flood points.

Figure 2: Vulnerability index of all nodes from graph with 1.4km radius.

Figure 3: SPAWN index of all nodes from graph with 1.4km radius.

Figure 4: Graph of flooding points in green with 1.4km radius.

ACKNOWLEDGMENTS

"Este trabalho foi realizado com o apoio da Loggi (www.loggi.com), por meio do Programa de Bolsas de Pesquisa Loggi (PBP-LOGGI)"

REFERENCES

[1] JHA, ABHAS K.; BLOCH, ROBIN; LAMOND, J. Cities and Flooding: a guide to integrated urban flood risk management for the 21st Century. The World Bank, Washington: 2012. DOI: 10.1596/978-0-8213-8866-2

[2] MARCELINO, E. V. DESASTRES NATURAIS e GEOTECNOLOGIAS e Conceitos Básicos. Inpe-Crs, p. 37, 2008.

[3] BESSA, ALINE; SANTOS, LEONARDO; MARTINEZ, LORENA; COSTA, MARIANA; CARDOSO, PEDRO; "Introdução às redes complexas" Universidade Federal da Bahia, Dezembro 2010, obtido em Janeiro de 2013 em: http://www.academia.edu/858124/Introducao_as_Redes_Complexas

[4] METZ, J. et al. Redes Complexas: Conceitos e Aplicações. Relatórios Técnicos do ICMC, p. 1–45, 2007.

[5] LEONARDO BL SANTOS, TANISHQ GARG, AURELIENNE AS JORGE, LUCIANA R LONDE, REGINA T REANI, ROBERTA B BACELAR, and IGOR M SOKOLOV. Vulnerability analysis in complex networks under a flood risk reduction point of view. arXiv, pages 1–9, 2020.

[6] V. GOLDSHTEIN, G. A. KOGANOV, and G. I. SURDUTOVICH. Vulnerability and Hierarchy of Complex Networks. pages 1–4, 2004.

[7] BOEING, G. OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. Computers, Environment and Urban Systems, v. 65, p. 126–139, 2017.

Uma abordagem baseada em clusterização hierárquica para resolver o Last-Mile Incremental Capacitated Vehicle Routing Problem (ICVRP)

Ruan Heleno Correa da Silva
rhcs@ic.ufal.br
Instituto de Computação, Universidade Federal de Alagoas
Maceió, Alagoas, BRA

Ascânio Sávio de Araujo Neves
asan2@ic.ufal.br
Instituto de Computação, Universidade Federal de Alagoas
Maceió, Alagoas, BRA

Rian Gabriel Santos Pinheiro
rian@ic.ufal.br
Instituto de Computação, Universidade Federal de Alagoas
Maceió, Alagoas, BRA

Bruno Costa e Silva Nogueira
bruno@ic.ufal.br
Instituto de Computação, Universidade Federal de Alagoas
Maceió, Alagoas, BRA

RESUMO

O objetivo desse projeto é resolver o *Last-Mile Incremental Capacitated Vehicle Routing Problem* (Last-Mile ICVRP) a partir da predição de modelos clusterizados organizados de maneira hierárquica. A hierarquia é constituída por dois níveis. No primeiro nível, o modelo inicial contém uma quantidade pré-definida k de clusters. No segundo nível, cada cluster $i \in \{1, \ldots, k\}$ tem M_i subclusters, tal que M_i é definido pelo algoritmo de distribuição das unidades de carregamento. A partir disso, cada subcluster do segundo nível será responsável por uma unidade de carregamento. Dessa forma, as entregas são encaminhadas para as unidades de carregamento a partir da predição de tais modelos. Nos resultados parciais obtidos, o algoritmo desenvolvido a partir dessa abordagem conseguiu bons resultados ao ser comparado com os resultados de outros algoritmos.

KEYWORDS

Clusterização, Last-Mile ICVRP

ACM Reference Format:
Ruan Heleno Correa da Silva, Ascânio Sávio de Araujo Neves, Rian Gabriel Santos Pinheiro, and Bruno Costa e Silva Nogueira. 2021. Uma abordagem baseada em clusterização hierárquica para resolver o Last-Mile Incremental Capacitated Vehicle Routing Problem (ICVRP). In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUÇÃO

O *Last-Mile Incremental Capacitated Vehicle Routing Problem* (Last-Mile ICVRP) pode ser descrito como se segue. Dado um conjunto (histórico) de clientes atendidos anteriormente $H = \{v_1, \ldots, v_\ell\}$, um conjunto $\mathcal{U}$ de unidades de carregamento e um conjunto com n novos clientes $D = \{v_{\ell+1}, \ldots, v_{\ell+n}\}$, o Last-Mile ICVRP tem como objetivo alocar cada uma das entregas do conjunto D a uma unidade de carregamento. Cada demanda é alocada, uma por vez, em uma única unidade de carregamento de acordo com a ordem de chegada pré-definida. As unidades de carregamento tem sua capacidade máxima que, quando atingida, precisa ser despachada para entrega. O problema busca achar uma alocação das entregas às unidades de carregamento, de maneira que a distância percorrida pelos veículos de entrega seja minimizada.

O *Last-Mile Incremental Capacitated Vehicle Routing Problem* (Last-Mile ICVRP) é um caso particular do problema SD-CVRP (Stochastic and Dynamic Capacitated Vehicle Routing Problems) [6]. O problema é denominado Last-Mile por caracterizar as entregas entre o centro de expedição e o destinatário final.

De maneira geral, a resolução dos problemas relacionados ao SD-CVRP pode ser dividida em duas etapas. A primeira etapa, chamada de *planejamento*, ocorre de maneira *offline* e utiliza dados históricos de entrega para computar um conjunto de decisões pré-definidas. A segunda etapa, chamada de *execução*, ocorre em tempo real e utiliza as ações pré-definidas na etapa anterior assim como as informações sobre os eventos em tempo real (ex.: chegada de uma nova demanda) para gerar as rotas finais.

O objetivo desse projeto é resolver o *Last-Mile Incremental Capacitated Vehicle Routing Problem* (Last-Mile ICVRP) usando uma estratégia baseada em clusterização hierárquica, organizada de maneira hierárquica em dois níveis.

2 TRABALHOS RELACIONADOS

Tradicionalmente, os VRP dinâmicos e estocásticos são formulados como processos de Markov [5, 8, 9] ou como programação estocástica [1]. Ambas as metodologias modelam os dados do problema como variáveis aleatórias que seguem uma distribuição previamente conhecida. O objetivo é otimizar uma medida de risco (como o valor esperado, a variância ou o valor em risco condicional de alguma função de custo), sujeito à satisfação de algumas restrições [3]. Note que ambas as abordagens não estão alinhadas com o problema encontrado na Loggi.

A partir de apresentações institucionais da empresa Loggi, é possível identificar duas estratégias recentes para resolver o Last-Mile ICVRP. A primeira estratégia foi apresentada em [2] e usa

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference'17, July 2017, Washington, DC, USA
© 2021 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

uma combinação de agentes autônomos com mineração de dados. Nesta abordagem, os agentes autônomos modelam as rotas a serem construídas. Além disso, técnicas de mineração de dados são usadas para extrair informações estocásticas sobre a distribuição de pacotes e assim otimizar a alocação de cada pacote em uma das rotas. Para determinar em qual rota r novo pacote p será alocado, a seguinte função é maximizada: $bet(p,r) = \alpha * bet_{DataMining}(p,r) + (1-\alpha) * bet_{Distance}(p,r)$, onde $bet_{DataMining}(p,r)$ é um valor extraído da mineração de dados, $bet_{Distance}(p,r)$ é uma função que mede a distância entre o pacote p e os pacotes já inseridos na rota r, e α é um valor no intervalo $[0,1]$. Os resultados mostram que o número de rotas que precisam ser geradas pela abordagem proposta é em média superior em 17% quando comparado com o número de rotas gerado por um solver para a versão estática do *CVRP*.

A segunda estratégia da empresa para resolver o Last-Mile ICVRP foi apresentada em video [7]. A partir do vídeo, é possível depreender que a estratégia consiste em duas etapas: uma etapa de clusterização rotas boas geradas a partir do histórico de entregas (*offline*), seguida de uma etapa alocação dos pacotes em um dos clusters de rotas (*online*). Três métodos de clusterização (*bag of cells*, distância euclidiana e distância entre ruas) e dois métodos de alocação de rotas (guloso e clusters artificiais) foram experimentados. A combinação do método de clusterização baseado em distância entre ruas com o de alocação usando clusters artificiais apresentou os melhores resultados. Estes resultados mostram que a solução apresentada é apenas 16% pior que as encontradas por meio de um solver para a versão estática do *CVRP*.

3 MÉTODO PROPOSTO

Para explicar o funcionamento do algoritmo desenvolvido, as principais funcionalidades foram divididas em cinco tópicos.

Figura 1: Fluxograma do método proposto.

A Figura 1 mostra o fluxograma do algoritmo desenvolvido utilizando uma abordagem baseada em Clusterização Hierárquica para resolver o *Last-Mile Incremental Capacitated Vehicle Routing Problem(ICVRP)*. Em vermelho, estão relacionados os processos referente à primeira etapa, chamada de *planejamento*, que ocorre de maneira *offline*. Em azul, estão relacionados os processos referente à segunda etapa, chamada de *execução*, que ocorre em tempo real.

3.1 Primeiro Nível da Clusterização Hierárquica

O primeiro nível é constituído pelo modelo de clusterização inicial. No algoritmo desenvolvido, é executado o algoritmo *K-Means* sobre as entregas com o valor pré-definido de k. O modelo é treinado a partir dos pontos das entregas presentes no histórico.

3.2 Distribuição das Unidades de Carregamento

A distribuição das Unidades de Carregamento tem como objetivo fornecer a quantidade de unidades de carregamentos para cada cluster. Ao utilizar as entregas do histórico e separar tais entregas em seus determinados clusters, a distribuição é feita de forma proporcional à quantidade de entregas presentes nesses clusters em relação a quantidade total de entregas. Com isso, multiplicamos cada percentual pelo número de unidades de carregamento. Por ser um número inteiro é necessário depois fazer o balanceamento para que a quantidade de unidades de carregamento que foram distribuídas seja igual ao número de unidades de carregamento inicial, além de buscar garantir que cada cluster tenha, no mínimo, uma unidade de carregamento. O balanceamento é feito removendo uma unidade de carregamento, por iteração, do primeiro cluster que tem mais de uma unidade de carregamento.

3.3 Segundo Nível da Clusterização Hierárquica

O segundo nível é constituído por k modelos de clusterização. A quantidade de clusters de cada modelo será o valor dado pelo algoritmo de distribuição de unidades de carregamento, de modo que cada modelo a ser criado será referente a um cluster do modelo inicial. A clusterização hierárquica busca tornar cada cluster dos modelos criados do segundo nível responsável por uma unidade de carregamento. Com isso, no algoritmo desenvolvido, cada modelo $i \in \{1, \ldots, k\}$ tem M_i subclusters, tal que M_i é definido pelo algoritmo de distribuição das unidades de carregamento, e com isso, é com as entregas do histórico referente àquele cluster.

3.4 Alocação das Entregas nas Unidades de Carregamento

A partir dos modelos, a análise de cada entrega será feita usando predições. Inicialmente é feita a predição da entrega a ser alocada no modelo do primeiro nível. O valor da predição da entrega no modelo do primeiro nivel será usado para identificar o modelo do segundo nível. Com isso, a predição do modelo do segundo nível será usada para identificar o cluster responsável pela unidade de carregamento.

No Algoritmo 1, é mostrado o pseudocódigo referente à alocação de entregas nas unidades de carregamento. A entrada contém um array com as entregas a serem alocadas, um array referente à distribuição, um array referente as unidades de carregamento, o modelo K-Means referente ao primeiro nível e o array de modelos K-Means referente ao segundo nível da clusterização hierárquica.

No algoritmo desenvolvido, aplicamos no ponto da entrega atual, a função *predict*, presente no algoritmo *K-Means*, que busca predizer o cluster mais próximo à um determinado ponto, no modelo do primeiro nível e no modelo do segundo nível, e assim identificamos o número referente a unidade de carregamento escolhida.

Após as predições, utilizamos um loop, que vai de 0 até o $u-1$, para identificar a posição da unidade de carregamento no array que

foi retornado pelo algoritmo de distribuição de unidades de carregamento. Após identificação, é feita a análise dos pesos já presentes naquela unidade de carregamento para saber se ao somar o peso da entrega, o valor total ultrapassa a capacidade. Caso ultrapasse, antes de adicionar a entrega atual, a unidade de carregamento contendo entregas já alocadas, é despachada. Depois dessa verificação, a entrega atual é alocada na determinada unidade de carregamento.

Algorithm 1 Pseudocódigo referente à alocação das entregas nas unidades de carregamento

```
Require: deliveries : [Delivery], D : [int], UCS : [UC], Model1 :
  KMeans, Models2 : [KMeans]
Ensure: routes ← [], u ← len(UCS)
  for delivery in deliveries do
      k ← Model1.predict(delivery.point)
      M ← Models2[k].predict(delivery.point)
      for i in range(0, u) do
          if k = D[i] then
              M ← M − 1
              uc ← i
          end if
          if M < 0 then
              break
          end if
      end for
      if (UCS[uc].size + delivery.size) > capacity_vehicles then
          routes.push(UCS[uc].deliveries)
          UCS[uc].pop(UCS[uc].deliveries)
      end if
      UCS[uc].push(delivery)
  end for
```

3.5 Organização das rotas

O despacho de uma unidade de carregamento acontece quando a capacidade máxima da mesma é atingido. Com isso, é executado o algoritmo que busca uma solução para o *TSP(Traveling Salesman Problem)* em cada conjunto de entregas. No fim, o mesmo processo de despacho acontece nas unidades de carregamento que não ultrapassaram a capacidade, executando o determinado algoritmo para organizar as rotas.

4 RESULTADOS PARCIAIS/PRELIMINARES

O algoritmo do projeto foi desenvolvido na linguagem de programação *Python*. Foi utilizado o método *K-Means*, da biblioteca *Sklearn*, como método de clusterização, além da utilização da biblioteca *Numpy*. Houve a instalação do servidor *OSRM* para utilizar as distâncias do *OpenStreetMap* e a aplicação do algoritmo implementado com a ferramenta *OR-Tools*, do *Google*, que busca resolver o *TSP* e *CVRP* disponibilizado pelo repositório *loggibud* [4], da *Loggi*, presente na plataforma *GitHub*.

Como forma de avaliação do algoritmo desenvolvido foi utilizado as instâncias cvrp, */train* e */dev*, dos diretórios *df-0*, *pa-0* e *rj-0* presentes no banco de dados disponibilizado pela *Loggi*. A pasta */train*, contém os dados que utilizamos como o histórico, distribuídos em 90 instâncias. A pasta */dev* é utilizada para gerar os resultados, de modo que ela contém 30 instâncias de testes.

O seguinte formato foi usado para encontrar um resultado a ser considerado geral de cada diretório:

(1) Inicialmente, separamos as entregas das instâncias de treino em batches (ou lotes) para que o algoritmo referente ao CVRP consiga encontrar uma solução. O tamanho do batch escolhido para os resultados foi 300, pois foi o tamanho que o algoritmo do arquivo */loggibud/v1/baselines/shared/ortools.py*, do repositório *loggibud*, consegue resolver em tempos razoáveis.
(2) Buscamos a distância percorrida das soluções encontradas para cada instância. As distâncias percorridas encontradas denominamos em resultados *off-line*.
(3) Executamos os algoritmos, de modo que a pegamos a distância percorrida das soluções geradas, utilizando a função *evaluate_solution*, presente no repositório *loggibud*, no arquivo */loggibud/v1/eval/task1.py*, e fazemos o cálculo da diferença em relação ao resultado obtido de cada instância do passo 1.
(4) Fazemos o cálculo da média dos percentuais, somando o cálculo da diferença encontrada entre o determinado algoritmo e o algoritmo que busca resolver o CVRP de cada instância (off-line) e dividimos pela quantidade de instâncias presente no diretório. O resultado do cálculo da média será considerado o resultado geral do diretório.

Como forma de comparação, utilizamos os resultados encontrados após a execução de dois algoritmos presentes no repositório *loggibud*, na pasta */loggibud/v1/baselines/task2*: *kmeans_greedy.py* e *qrp_sweep.py* com os valores de k no intervalo de 4 a 10.

4.1 Diretório Pará (pa-0)

O diretório pa-0 contém mais de 25000 entregas em seu histórico. Por ter uma média próxima de 300 entregas por instância, consideramos esse diretório como um diretório com pequenas instâncias, em comparação com os outros dois diretórios utilizados. Para os testes, foram 14 unidades de carregamento no algoritmo referente à Clusterização Hierárquica.

Tabela 1: Resultados encontrados do diretório pa-0

N_clusters	C_Hierárquica	KMeans_greedy	QRP_sweep
4	18.50%	**16.34%**	41.30%
5	**14.96%**	17.22%	24.61%
6	21.99%	23.89%	33.68%
7	21.55%	23.17%	34.86%
8	22.22%	22.44%	30.91%
9	24.93%	25.34%	25.99%
10	26.29%	26.72%	**20.79%**

A Tabela 1 apresenta os resultados referentes a instância pa-0. Nesta tabela, as colunas 2, 3, 4. representam os resultados, para cada quantidade de cluster, dos métodos: *clusterização hierárquica*, *kmeans_greedy.py* e *qrp_sweep.py*. Em negrito, destacamos os melhores resultados de cada método.

Figura 2: Comportamento dos resultados encontrados do diretório pa-0.

A Figura 2 mostra o comportamento dos resultados encontrados de pa-0 dos três métodos no intervalo de número de clusters utilizado.

4.2 Diretório Distrito Federal (df-0)

O diretório df-0 contém mais de 88000 entregas em seu histórico. Por ter uma média próxima de 1000 entregas por instância, consideramos esse diretório como um diretório com instâncias médias, em comparação com os outros dois diretórios utilizados. Para os testes, foram 14 unidades de carregamento no algoritmo referente à Clusterização Hierárquica.

Figura 3: Aplicação da Clusterização Hierárquica com dois níveis de clusterização.

A Figura 3 apresenta um exemplo de aplicação utilizando as entregas do histórico de df-0. Clusterização com 6 clusters no primeiro nível, e 28 clusters no segundo nível distribuído entre os 6 modelos.

Tabela 2: Resultados encontrados do diretório df-0

clusters	C_Hierárquica	KMeans_greedy	QRP_sweep
4	16.94%	18.21%	25.56%
5	14.19%	17.22%	20.53%
6	**10.55%**	**12.04%**	23.30%
7	12.95%	15.41%	19.92%
8	18.60%	16.88%	18.19%
9	18.40%	16.92%	**18.09%**
10	17.09%	17.68%	21.56%

De forma similar à Tabela 1 e à Figura 2, temos a Tabela 2 e a Figura 4, respectivamente. Eles apresentam os resultados encontrados em relação ao algoritmo off-line para cada número de cluster nas instâncias e seu comportamento no intervalo de 4 a 10 clusters.

Figura 4: Comportamento dos resultados encontrados do diretório df-0.

4.3 Diretório Rio de Janeiro (rj-0)

O diretório rj-0 contém mais de 320000 entregas em seu histórico. Por ter uma média próxima de 3500 entregas por instância, consideramos esse diretório como um diretório com instâncias grandes, em comparação com os outros dois diretórios utilizados. Para os testes, foram 28 unidades de carregamento no algoritmo referente à Clusterização Hierárquica.

Tabela 3: Resultados encontrados do diretório rj-0

N_clusters	C_Hierárquica	KMeans_greedy	QRP_sweep
4	17.43%	25.67%	23.56%
5	16.82%	25.17%	23.87%
6	**16.72%**	24.93%	21.46%
7	17.54%	23.87%	21.44%
8	17.33%	**23.00%**	20.47%
9	18.34%	23.86%	**19.31%**
10	18.09%	23.56%	19.80%

Figura 5: Comportamento dos resultados encontrados do diretório rj-0.

De forma similar à Tabela 1 e à Figura 2, temos a Tabela 3 e a Figura 5, respectivamente. Eles apresentam os resultados encontrados em relação ao algoritmo off-line para cada número de cluster nas instâncias e seu comportamento no intervalo de 4 a 10 clusters.

5 CONCLUSÃO

Neste projeto trouxemos uma abordagem baseada em clusterização hierárquica com o objetivo de ser uma alternativa para resolver o *Last-Mile Incremental Capacitated Vehicle Routing (Last-Mile ICVRP).*

Tal abordagem se fez presente a partir da aplicação do algoritmo de distribuição das unidades de carregamento que tornou possível a criação dos modelos que constituem o segundo nível da hierarquia. Assim como há a dependência de tal distribuição para o andamento do algoritmo, a quantidade de unidades de carregamento também interfere nos resultados. Ao analisar os resultados parciais obtidos, percebemos que os melhores resultados do algoritmo desenvolvido encontrados em cada diretório são mais satisfatórios que os melhores resultados dos algoritmos *KMeans_greedy.py* e *QRP_Sweep.py*. Podendo, assim, concluir que tal abordagem é válida para aplicação.

6 TRABALHOS EM ANDAMENTO E FUTUROS

Na análise do desenvolvimento do algoritmo, entre os possíveis próximos trabalhos, pode-se relacionar:

(1) Adição de novos níveis na clusterização hierárquica.
(2) Desenvolvimento de uma nova forma de distribuição das unidades de carregamento.

Em relação aos trabalhos em andamento, está sendo estudado uma maneira de simplificar a criação e o treinamento dos modelos do segundo nível da clusterização hierárquica.

REFERÊNCIAS

[1] Dimitris Bertsimas, Philippe Chervi, and Michael Peterson. 1995. Computational approaches to stochastic vehicle routing problems. *Transportation science* 29, 4 (1995), 342–352.

[2] Juan Camilo Fonseca-Galindo, Gabriela de Castro Surita, José Maia Neto, Cristiano Leite de Castro, and André Paim Lemos. 2020. A Multi-Agent System for Solving the Dynamic Capacitated Vehicle Routing Problem with Stochastic Customers using Trajectory Data Mining. *arXiv preprint arXiv:2009.12691* (2020).

[3] Chrysanthos E Gounaris, Panagiotis P Repoussis, Christos D Tarantilis, Wolfram Wiesemann, and Christodoulos A Floudas. 2016. An adaptive memory programming framework for the robust capacitated vehicle routing problem. *Transportation Science* 50, 4 (2016), 1239–1260.

[4] Loggi. 2021. loggiBUD: Loggi Benchmark for Urban Deliveries. https://github.com/loggi/loggibud.

[5] Warren B Powell. 1988. A Comparative review of alternative algorithms for the dynamic vehicle allocation problem. In *Vehicle routing: methods and studies. (Studies in Management Science and Systems, Volume 16)*, B. Golden and A. Assad (Eds.). Amsterdam ; New York : North-Holland, 249–291.

[6] Ulrike Ritzinger, Jakob Puchinger, and Richard F Hartl. 2016. A survey on dynamic and stochastic vehicle routing problems. *International Journal of Production Research* 54, 1 (2016), 215–231.

[7] Gabriela Surita and José C. Maldonado. 2020. Evolução das Soluções e Experiência de Roteirização na Loggi. https://www.youtube.com/watch?v=IDmaN7ima7k. [Online; acessado em 28 Março 2021].

[8] Barrett W Thomas and Chelsea C White Iii. 2004. Anticipatory route selection. *Transportation Science* 38, 4 (2004), 473–487.

[9] Marlin W Ulmer, Justin C Goodson, Dirk C Mattfeld, and Marco Hennig. 2019. Offline–online approximate dynamic programming for dynamic vehicle routing with stochastic requests. *Transportation Science* 53, 1 (2019), 185–202.

Solução para o Problema de Roteamento de Veículos com Janela de Tempo Aplicado aos Conjuntos de Entradas do Benchmark Loggi BUD

Élani Marques Zanlucas
Faculdade de Computação - FACOM
Universidade Federal de Mato Grosso do Sul - UFMS
Campo Grande, MS, Brasil
elani.m.zanlucas@ufms.br

Rúbia Mara de Oliveira Santos
Instituto de Matemática - INMA
Universidade Federal de Mato Grosso do Sul - UFMS
Campo Grande, MS, Brasil
rubia.oliveira@ufms.br

ABSTRACT

Este projeto propõe aplicar o Algoritmo Simplex com Geração de Colunas para resolução do Problema de Roteamento de Veículos com Janela de Tempo utilizando o conjunto de dados de teste e validação do benchmark Loggi BUD. A abordagem proposta utiliza um algoritmo de Caminho Mínimo com Janela de Tempo baseado na generalização de Ford-Bellman-Moore. Esse algoritmo surge como sub-problema no algoritmo Primal Simplex para resolver a relaxação linear do problema de partição de conjuntos utilizado na modelagem do roteamento com janelas de tempo.

KEYWORDS

Roteamento com Janela de Tempo, Geração de Colunas, Caminho Mínimo.

ACM Reference Format:
Élani Marques Zanlucas and Rúbia Mara de Oliveira Santos. 2021. Solução para o Problema de Roteamento de Veículos com Janela de Tempo Aplicado aos Conjuntos de Entradas do Benchmark Loggi BUD. In *Proceedings of Workshop Resultados do PBP-Loggi (Workshop).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUÇÃO

A roteirização de veículos é uma das histórias de grande sucesso da Pesquisa Operacional nas últimas décadas. Isto pode ser medido pelo expressivo número de artigos que vem sendo publicados ao longo dos anos na literatura especializada. O Problema de Roteamento de Veículos desempenha um papel central no gerenciamento da distribuição de bens e serviços. Sua importância é refletida na grande variedade de aplicações, as quais incluem o roteamento de ônibus urbano, sistemas de coleta de lixo, entrega de periódicos a assinantes, entre outros [3, 4]. Dependendo da aplicação, há necessidade de alterar as restrições do problema. Algumas restrições bem conhecidas para o problema de roteamento de veículos são: restrições de horário de atendimento (conhecidas na literatura como janelas de tempo ou janelas horárias), capacidades dos veículos, frota de veículos de diferentes tamanhos, duração máxima dos roteiros dos veículos (tempo ou distância) e restrições de veículos que podem atender determinados clientes. O setor de logística é aquele que tem o maior interesse em soluções eficientes para esses tipos de problemas, uma vez que há demandas por aplicações específicas, como entregas na menor rota possível, ou mesmo por combinações desses problemas, como entregas dentro de janelas de tempo restritas, com veículos de diferentes capacidades e com possibilidade de inclusão de novas rotas.

Este projeto de Iniciação Científica consiste em estudar e aplicar um método baseado na técnica de geração de colunas para resolução do Problema de Roteamento de Veículos com Janelas de Tempo (PRVJT). Especificamente, a partir do trabalho [4], vislumbra-se utilizar o algoritmo Simplex com Geração de Colunas em conjunto com o algoritmo de Caminho Mínimo com Janelas de Tempo (CMJT) baseado na generalização do algoritmo clássico de Ford-Bellman-Moore [2, 5]. A utilização do algoritmo CMJT permite resolver o problema de partição de conjuntos, utilizado na modelagem do problema de roteamento com janelas de tempo. A adoção da técnica de Geração de Colunas possibilita obter subproblemas que permanecem com características combinatórias semelhantes ao problema original. Como infraestrutura para testes, serão feitas adaptações nas instâncias de entradas fornecidas pelo benchmark Loggi BUD para validação da solução.

2 DESCRIÇÃO DO PROBLEMA

2.1 Problema de Roteamento de Veículos

O Problema de Roteamento de Veículos (PRV), sendo uma generalização do Problema do Caixeiro Viajante, foi introduzida em 1959 por Dantzig e Ramser com o artigo "The Truck Dispatching Problem". O PRV tem como objetivo definir um conjunto de rotas a serem percorridas por uma frota de veículos a um custo mínimo, obedecendo as seguintes restrições[1]:

- 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.

A Figura [1] apresenta o conceito do PRV por meio de um grafo orientado, onde os veículos partem do nó $d1$, fazem suas respectivas rotas (representadas pelas arestas), passando pelos clientes (nós contidos nas rotas) somente uma vez e retornam ao nó inicial.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Workshop, Dezembro 08, 2021, Brasil
© 2021 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Considere um grafo orientado completo $G = (V, E)$, onde $V = \{0, 1, \cdots, n\}$ é o conjunto de vértices de G e $E = \{(i, j) : i, j \in V, i < j\}$ é o conjunto de arcos de G. O vértice 0 de V representa o depósito e os vértices restantes representam os clientes. Cada cliente i possui uma demanda não-negativa q_i conhecida. Existem k veículos homogêneos de capacidade Q e cada arco $(i; j)$ está associado a um custo não-negativo c_{ij} de deslocamento que parte do cliente i ao cliente j, tal qual um tempo de viagem t_{ij}, que será desmembrado como as Janelas de Tempo, sendo o tempo de serviço do cliente i. Em suma, o PRV consta em determinar um conjunto de rotas, de forma que cada cliente seja visitado apenas uma vez por um único veículo, tal que o custo seja mínimo e a demanda atendida. Ademais, cada veículo inicia e termina o percurso no vértice 0, a demanda total de cada rota não deve superar a capacidade Q do veículo e o tempo de viagem não pode exceder o limite D. Assim, as variáveis são associadas como:

- $x_{i,j,k} = \begin{cases} 1 & \text{se o veículo } k \text{ visita o cliente } j \text{ após o cliente } i \\ 0 & \text{caso contrário.} \end{cases}$

Logo, temos a seguinte modelagem:

$$Min \quad \sum_{k=1}^{K} \sum_{j=1}^{n} c_{i,j} x_{i,j,k}$$

$sa:$

$$\sum_{k=1}^{K} \sum_{j=1}^{n} x_{i,j,k} = 1, \ \forall i = 1, 2, \cdots, n \tag{1}$$

$$\sum_{i=1}^{n} q_i \sum_{j=1}^{n} x_{i,j,k} \leq Q, \ \forall k = 1, 2, \cdots, K \tag{2}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} t_{i,j} x_{i,j,k} \leq D, \ \forall k = 1, 2, \cdots, K \tag{3}$$

$$\sum_{j=1}^{K} x_{0,j,k} = 1, \ \forall k = 1, 2, \cdots, K \tag{4}$$

$$\sum_{i=0}^{n} x_{i,h,k} - \sum_{j=0}^{n} x_{h,j,k} = 0, \ \forall h = 1, ..., n, \ \forall k = 1, ..., K \tag{5}$$

$$\sum_{i=1}^{n} x_{i,n+1,k} = 1, \ \forall k = 1, 2, \cdots, K \tag{6}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} x_{i,j,k} \leq |S| - 1, \ S \subset C, 2 \leq |S| \leq \frac{n}{2}, k = 1, ..., K \tag{7}$$

$$x_{i,j,k} \in \{0, 1\}, \ i = 1, \cdots, n; \ j = 0, \cdots, n; \ k = 1, \cdots, m; \tag{8}$$

A restrição (1) assegura que cada veículo saia do depósito e chegue a um certo cliente. A restrição (2) impõe que a demanda total de cada rota do veículo k não supere a capacidade Q do veículo. A restrição (3) garante que o intervalo de tempo de cada rota do veículo k não exceda o limite D. As restrições (4), (5) e (6) descrevem restrições de fluxos em rede, os quais demandam que cada veículo k parta do nó (0) uma única vez, deixem o nó h se, e somente se, entrar neste nó, e retorne ao depósito (nó $n + 1$) apenas uma vez. A restrição (7) certifica a não existência de sub-rotas. A restrição (8) afirma que as variáveis de decisão x_{ijk} são binárias.

2.2 Problema de Roteamento de Veículos com Janela de Tempo

As Janelas de Tempo são intervalos de tempo a serem respeitados, sendo restrições importantes para determinar a eficiência da entrega de um produto. Estes intervalos podem ser descritos como horário de atendimento. Sendo assim, o Problema de Roteamento de Veículos com Janela de Tempo trata-se de estabelecer a distância das rotas a serem percorridas por uma frota de veículos a um custo mínimo, levando em consideração o intervalo limite de tempo para a circulação destes veículos. De acordo com [4], temos a seguinte modelagem para o PRVJT:

Considere um conjunto de viagens onde cada viagem i é especificada por um intervalo de tempo $[a_i; b_i]$, no qual a viagem deve ser iniciada. Uma viagem é chamada uma jornada produtiva. Define-se inter-viagem ou viagem de conexão como sendo uma jornada não produtiva conduzida por um veículo. Assim, uma inter-viagem é representada pelo arco $(i; j)$ que vai do final da viagem i para o início da viagem j. A cada inter-viagem $(i; j)$ associamos um tempo de duração dado por t_{ij} e um custo dado por c_{ij}. Uma rota é uma sequência de viagens e inter-viagens conduzida pelo mesmo veículo. Considere P como sendo o conjunto das viagens, I o conjunto das inter-viagens, A o conjunto de arcos (arestas), $A = I \cup (\{s\} \times P) \cup (P \times \{t\})$, e N o conjunto de nós, $N = P \cup \{s, t\}$. Uma interviagem é considerada se e somente se for possível realizar a viagem j depois da viagem i, onde a restrição de intervalo de tempo $(a_i + t_{ij} \leq b_j)$ é respeitada. A formulação para a definição de rotas ótimas para viagens pode ser definida a partir das seguintes variáveis:

- $x_{ij} = \begin{cases} 1 & \text{se o arco } (i, j) \in A, \text{ for usado por uma conexão} \\ 0 & \text{caso contrário.} \end{cases}$

- t_i: variável que representa o tempo associado ao início de cada nó i, com $i \in R$.

As rotas ótimas que respeitam as restrições de horários são as soluções para o seguinte problema:

$$
\begin{aligned}
&Minimizar && \sum_{(i,j)\in A} c_{ij}x_{ij} \\
&sujeito\ a: && \\
& && \sum_{j\in N} x_{ij} = 1,\ i \in P && (8)\\
& && \sum_{k\in N} x_{ik} = 1\ i \in P && (9)\\
& && x_{ij} \geq 0; \qquad (i,j) \in A && \\
& && x_{ij} > 0 \Rightarrow t_i + t_{ij}; \leq t_j \qquad (i,j) \in N && (10)\\
& && a_i \leq t_i \leq b_i; \qquad i \in R && (11)\\
& && x_{ij} = \{0,1\} \qquad (i,j) \in A &&
\end{aligned}
$$

A restrição (8) informa que de algum nó j a conexão chegará ao nó i e a restrição (9) afirma que a conexão que chegou ao nó i vai ter que partir para algum nó j. A restrição (10) descreve a compatibilidade requerida entre as rotas e os horários e a restrição (11) estabelece o intervalo de tempo no qual a conexão deve ser iniciada.

3 ALGORITMOS

3.1 Simplex com Geração de Colunas

3.1.1 Método Simplex. O método inicia-se com uma solução básica factível e busca apenas outras soluções básicas melhores a fim de encontrar uma solução ótima factível, de acordo com o critério de otimalidade estabelecido.

3.1.2 Algoritmo Simplex.

- Fase I: inicialização de uma partição básica factível
- Fase II: iteração pelo Método Simplex

3.1.3 Geração de Colunas. Também conhecida como método de Dantzig-Wolfe, a geração de colunas é um método de otimização discreta que possui como conceito os pontos extremos e os raios extremos, os quais geram qualquer ponto de um poliedro, teoricamente gerando uma "borda" ao poliedro. Para este método, é aplicado o Princípio da Decomposição, isto é, resolução de subproblemas menores, já que o problema original, conforme o tamanho aumenta, se torna NP-Difícil. Com relação ao problema estudado (PRVJT), as colunas são as rotas a serem geradas.

3.1.4 Algoritmo Simplex com Geração de Colunas. A geração de colunas é aplicada para determinar subconjuntos de colunas a serem utilizados como base factível, a fim de melhorar a solução do Método Simplex.

3.2 Ford-Bellman-Moore

O algoritmo de Ford-Bellman-Moore para o problema de caminho mínimo (sem janela de tempo), determina para cada nó uma etiqueta c_i o qual representa o custo de ir da origem s para o nó i utilizando a rota de custo mínimo. Os rótulos são melhorados recursivamente até que os valores ótimos sejam obtidos.

Seja $F(i) = \{j : (i,j) \in A, j \in \mathbb{N}\}$ o conjunto de sucessores do nó i e L o conjunto de nós que ainda não foram explorados. A árvore de caminho mínimo saindo da origem o para todos os outros nós da rede é obtida associando-se a cada nó uma etiqueta que é melhorada a cada iteração do algoritmo. O método das etiquetas consiste na escolha de um nó i pertencente ao conjunto L, em seguida retira-se o nó i do conjunto L e tenta-se melhorar todos os sucessores do nó i. Logo, o novo nó melhorado é inserido no conjunto L. Assim, quando o conjunto L for vazio, a etiqueta c_i associada ao nó i representará o caminho de comprimento mínimo saindo da origem s e terminando no nó i.

No problema de Caminho Mínimo, o algoritmo determina uma etiqueta θ_i para cada nó, a qual representa o custo de sair da origem e chegar ao nó i pela rota de custo mínimo e que é melhorada a cada iteração do algoritmo. Para o caso de estudo, a etiqueta trata-se de um vetor com tempo e custo associados. Com isso, seja $Q_t = (t_i^\alpha, c_i^\alpha), \alpha = 1, 2, ..., \alpha(i)$ um conjunto irredutível de etiquetas para o nó i. Para obtenção do conjunto Q_j, associado ao nó j, unimos o arco (i,j) aos caminhos que partem da origem e chegam em i com a seguinte aplicação:

$$F_{ij}(Q_i) = \cup_\alpha f_{ij}\{(t_i^\alpha, c_i^\alpha)\}$$

3.3 Natureza das Etiquetas

Para resolvermos o problema de caminho mínimo com janela de tempo, as rotas têm outro atributo importante que consiste no tempo de chegada até ao nó i. Então, para tratarmos simultaneamente as variáveis de tempo t_i e as variáveis de custo c_i associadas a cada nó i, utilizaremos uma etiqueta formada por (t_i, c_i), para rotular cada nó i. Uma etiqueta (t_i, c_i) é atribuída ao nó i se, e somente se, existir um caminho possível de custo c_i, saindo da origem s e chegando ao nó i no menor tempo t_i. Qualquer caminho saindo da origem s e chegando ao nó i com custo c_i e com o tempo pertencente ao intervalo $[t_i, b_i]$ não tem custo de espera.

No algoritmo de Ford-Bellman-Moore, o problema clássico do Caminho Mínimo, associa a cada nó uma única etiqueta que é progressivamente melhorada. Quando acrescentamos ao problema as restrições de horários (janela de tempo) não é mais possível conservar unicamente a etiqueta melhorada para cada nó, porque o par (t_i, c_i) não tem ordem total no $\Re^2$: então é necessário preservar um conjunto de etiquetas (t_i^α, c_i^α), $\alpha \geq 1$. Este conjunto é finito porque a quantidade de nós é finita, determinando um número finito de caminhos possíveis saindo da origem o e chegando ao nó i, e cada caminho produz uma etiqueta. Estas etiquetas formam para cada nó i uma lista Q_i. A lista Q_i é atualizada sequencialmente a partir das informações em Q_i, juntamente às informações encontradas ao considerar os caminhos de s e j, compostas do caminho de s e i do arco (i,j).

3.4 Relação de Ordem das Etiquetas

Construímos um conjunto de etiquetas Q_i para cada nó i que guarda as menores distâncias referentes aos diferentes tempos localizados no intervalo $[a_i, b_i]$. Definimos então a relação de ordem parcial $\prec$ para o nó i:

$$(t_i^1, c_i^1) \prec (t_i^2, c_i^2) \Leftrightarrow (t_i^1 \leq t_i^2) \text{ e } (c_i^1 \leq c_i^2).$$

Uma etiqueta pertencente ao conjunto Q_i é denominada *etiqueta mínima* se não existir um outra etiqueta inferior a ela segundo uma relação de ordem. Assim, as etiquetas mínimas guardam o comprimento do caminho ótimo da origem o ao destino i. Um conjunto de etiquetas é dito irredutível se todas as etiquetas são mínimas. Uma etiqueta mínima é *ótima* (t_i^*, c_i^*) se, quando aplicarmos o princípio de otimalidade (se conhecemos os caminhos mínimos de todos os predecessores de j, então, podemos determinar o caminho mínimo até j) a esta etiqueta, se ela continuar com o menor caminho satisfazendo as restrições de horário em relação as outras etiquetas do conjunto. O resultado desta aplicação permite reduzir o conjunto de etiquetas de cada nó, retirando aquelas que não são mínimas e conservando o conjunto das etiquetas irredutíveis dentro do tempo de atendimento permitido. O conjunto irredutível associado ao nó i é ordenado em função da variável de tempo que é estritamente crescente, observando que os comprimentos c_i estão em ordem decrescente.

O conjunto irredutível associado ao nó i contém $\alpha(i)$ pares de etiquetas: (t_i^α, c_i^α), $\alpha = 1, 2, \ldots, \alpha(i)$, com

$$a_i \leq t_i^1 < t_i^2 < \cdots < t_i^{\alpha(i)} \leq b_i \quad \text{e} \quad c_i^1 > c_i^2 > \cdots c_i^{\alpha(i)}.$$

3.5 Formulação por Partição de Conjuntos

3.6 Problema-Mestre

Tem-se como objetivo resolver a formulação das rotas a serem utilizadas. Por conta disso, aplica-se o Algoritmo Simplex com Geração de Colunas, resolvendo a relaxação linear. Pelo algoritmo, são disponibilizados os multiplicadores simplex para geração de colunas.

3.7 Sub-Problema

O algoritmo de Caminho Mínimo com Janelas de Tempo (CMJT) empregado no sub-problema, é uma adaptação do Algoritmo de Ford-Bellman-Moore. Neste caso, o CMJT retorna uma rota de custo mínimo que satisfaz a janela de tempo estipulada. Assim como o FBM, este algoritmo determina uma etiqueta para representar o custo da origem ao destino i. Como há o diferencial da restrição do tempo, o algoritmo CMJT armazena para cada nó i um conjunto de pares de etiquetas $Q_i = (t_i^\alpha, \theta_i^\alpha)$, onde $\alpha \geq 1$. Em outras palavras, o conjunto Q_i é representado pelo tempo de chegada ao nó i e o custo da rota i. Um exemplo mais prático desta etiqueta, é o funcionamento do GPS, o qual é dado um ponto de origem e um ponto de destino e baseado nisso, são calculadas várias rotas com seus respectivos tempos e distâncias.

4 RESULTADOS PRELIMINARES

4.1 Benchmark Loggi BUD

Um ponto de relevância notado para este projeto reside em dar continuidade ao trabalho [4], que apresentou um método interessante para o Problema de Roteamento de Veículos com Janela de Tempo. Experimentos e testes foram executados com esse método utilizando conjuntos de entradas baseados nas instâncias de Solomon com até 100 nós, concluindo que a aplicação da técnica é atrativa, pois gerou ganhos de desempenho significativos nos problemas testados. Nesse contexto, um desafio direto desta pesquisa é avaliar se a técnica proposta é robusta e escalável para instâncias da ordem de milhares de pontos de entrega, como é o caso dos conjuntos de dados do benchmark Loggi BUD. Ademais, outro ponto relevante da proposta é notar a frequente demanda por entregas dentro de janelas de tempo bem específicas. No cenário de logística brasileiro, diferentes clientes (órgãos públicos, comércio, indústria, residências, condomínios) podem ter disponibilidades e restrições de horários distintos, o que força, diretamente, que esse tipo de restrição seja considerado na modelagem e solução do problema de roteamento de veículos.

Como o benchmark Loggi BUD não disponibiliza Janelas de Tempo, foi preciso encontrar uma maneira de manipular (controlar) o tempo a partir do OSRM. Ao calcularmos a distância, é possível saber o tempo do percurso e com isso, podemos estimar as Janelas de Tempo. Aqui encontramos a maior dificuldade.

Para o uso do Benchmark, foi feito um clone do git loggibud, assim como foram baixados e extraídos os datasets disponibilizados pela Loggi e o OSRM dentro desta pasta. Com isso, temos uma pasta denominada "data", que possui os dados para a seleção de um subconjunto pequeno de instância, baseado na documentação do git da loggibud. Para propósito de treino, foram disponibilizadas instâncias do Distrito Federal, Pará e Rio de Janeiro.

4.2 OSRM

A Open Source Routing Machine é um mecanismo de roteamento para encontrar caminhos mais curtos nas redes rodoviárias. Assim como o Google Maps, o OSRM é uma forma de obter cálculo de distâncias, sendo um diferencial por ser uma plataforma open source. O código fonte está no github, assim como um Readme e a Documentação do OSRM. Para utilizá-lo, é preciso criar uma imagem no Docker, disponível pela própria plataforma, o qual há dois pipelines pré-processados: Hierarquia de Contrato e Dijkstra Multi-Nível.

4.3 Dificuldades Encontradas

Infelizmente, não foi possível apresentar resultados computacionais devido à dificuldade encontrada na manipulação de diversos algoritmos simultaneamente. Além disso, também encontramos dificuldades em incluir janelas de tempo no benchmark Loggi BUD, para testar o método proposto. Por fim, selecionamos um subconjunto para efetuar os testes, como essa decisão foi tardia, ainda não estão finalizados os dados de entradas.

Como trabalhos futuros pretendemos focar no aprimoramento dos algoritmos implementados a fim de obter boas soluções para o benchmark. Existe uma possibilidade de aprofundar os estudos em torno da paralelização dos algoritmos implementados.

REFERENCES

[1] Marcos Arenales, Vinícius Armentano, Reinaldo Morabito, and Horacio Yanasse. 2011. *Pesquisa Operacional para Cursos de Engenharia.* ABEPRO.
[2] Leiserson. C. E. Cormen. T. H. and Rivest. R. L. 1990. Introduction to Algorithms. *MIT Press* (1990).
[3] C. B. Cunha. 1997. Methods for Routing with Time Windows. *Tese de Doutorado, USP, Sao Paulo* (1997).
[4] Rúbia M. Oliveira and Ricardo R. Santos. 2008. Problema de Roteamento com Janelas de Tempo: Uma Abordagem via Geracao de Colunas. *Simposio Brasileiro de Pesquisa Operacional* (2008), 1802–1812.
[5] R. E. Tarjan. 1983. Data Structures and Network Algorithms. *Conference Series in Applied Mathemathics* (1983).

Last-Mile Capacitated Vehicle Routing Problem

Vinicius G. A. V. Resende
Department of Computing
Federal University of Ouro Preto
Ouro Preto, Minas Gerais, Brazil
vinicius.verona@aluno.ufop.edu.br

Puca H. V. Penna
Department of Computing
Federal University of Ouro Preto
Ouro Preto, Minas Gerais, Brazil
puca@ufop.edu.br

Túlio A. M. Toffolo
Department of Computing
Federal University of Ouro Preto
Ouro Preto, Minas Gerais, Brazil
toffolo@ufop.edu.br

ABSTRACT

Despite academic efforts on solving vehicle routing problems (VRP), many real-world scenarios remain very challenging in terms of computational resources. Some examples of such challenging problems can be seen in Loggi Benchmark for Urban Deliveries (LoggiBud) instances comprehending three major Brazilian cities: *Rio de Janeiro*, *Brasília* and *Belém*. This manuscript presents preliminary results obtained during a 7-month initiation research program focused on the real-life package delivery problems proposed by LoggiBud. The Iterated Local Search (ILS) metaheuristic is used to explore the search space with two inter-route neighborhood moves – Swap and Insert – and an extension of Lin-Kernighan heuristic, responsible for intra-route optimization. As shown by preliminary experiments, the proposed algorithm integrating ILS and the Lin-Kernighan heuristic is capable of considerably improving solutions obtained by the classical Clarke-Wright constructive heuristic.

KEYWORDS

Loggi Benchmark for Urban Deliveries, Vehicle Routing Problem (VRP), Iterated Local Search (ILS), Lin-Kernighan Heuristic (LKH)

1 INTRODUCTION

The Vehicle Routing Problem (VRP) can be defined as an undirect weighted graph $G(V, E)$ where each vertex is a location and each edge is a path connecting two vertices [7]. Generally, a VRP contains a symmetric matrix $C = [c_{i,j}]$ corresponding to the traveling cost (distance) from i to j along a weighted edge, where the cost matches the weight.

The Capacitated Vehicle Routing Problem (CVRP) is a VRP extension which includes capacity constraints. The problem considers a set of available vehicles that leave from a depot to attend groups of customers, each with different demands. The most usual objective is to minimize the number of vehicles used as well as the total distance traveled.

Since its introduction, numerous techniques and strategies have been proposed to address the problem. These techniques are generally separated into two categories: exact methods and heuristic methods. Oyola et al. [6, 7] present an overview about CVRP extensions, such as stochastic versions of problem, and propose techniques in both mentioned categories, as well as hybrid approaches. Some of the techniques presented are: a new *Dantzig-Wolfe* (DW) version used on a Capacitated Vehicle Routing Problem with Stochastic Demand (CVRPSD) where a customer with multiple visits and non-elementary routes are both accepted; *Adaptive Large Neighborhood Search* (ALNS), *Tabu Search* (TS), *Simulated Annealing* (SA), *Iterated Local Search* (ILS), and *Dynamic Programming*. More recently, Christiaens and Vanden Berghe [2] proposed a ruin-and-recreate based algorithm capable of obtaining very impressive. However, as stated by the authors, despite being actively studied since its introduction in 1959 by Dantzig and Ramser, the CVRP continues to prove itself a very significant challenge.

Several decomposition strategies were also proposed and evaluated for the CVRP. Toffolo et al. [11], for instance, evaluated different strategies for decomposing the problem into assignment and routing, obtaining different search spaces and resulting in many improving solutions. Uchoa et al. [12] proposed a branch-and-cut-and-price algorithm which may be seen as the state-of-the-art exact solver for the CVRP. They authors were able to solve many previously unsolved CVRP instances. These approaches, however, were never evaluated on very large CVRP with more than 7000 customers such as some of the instances introduced by Loggi [4].

This research considers a CVRP with an unlimited set of available vehicles with identical capacity n based on the same depot. These vehicles must distribute k products to specific points within a city, with k ranging from 224 to 7094 packages. All data used and mentioned henceforth has been provided by Loggi [4] in *JavaScript Object Notation* (JSON) for the following Brazilian major cities: *Rio de Janeiro*, *Brasília* and *Belém*.

To evaluate each solution, basically one criteria is taken into consideration: the total distance traveled by the vehicles in a given solution. As stated in Section 2.6, the objective is to minimize the total distance.

From this section on, this article is organized as follows: first, Section 2 details the scenario's pre-processing phase, the techniques used to solve the problem, as well as the evaluation function for a solution. Then, experiments and preliminary results are provided by Section 3. Finally, conclusions are drawn and future work is presented by Section 4.

2 PROPOSED METHODOLOGY

To facilitate understanding the methodology, the section is divided into three main parts. Reading, organizing, and processing the input data are detailed in the first part. In the second and third parts, the initial solution algorithm and the optimization phase are detailed, respectively.

Before proceeding, note the difference between a route and a feasible solution. A route is a path traveled by a single vehicle, starting and ending at a depot/hub. Consequently, a set of routes is a feasible solution if each route is limited to the vehicle capacity. Each demand must be served only once in a solution. Furthermore, it is noteworthy that this article defines a customer and a delivery as the same thing.

Workshop PBP-Loggi, December 2021, São Paulo, Brazil
© 2021

2.1 Reading and Pre-processing

As previously stated in Section 1, all input data used was provided by Loggi [4] in JSON format. A webservice is used to compute distances between each pair of deliveries. To enable quick reading of instances and avoid *HTTP* requests, the distance matrix is computed once using an *HTTP* request to LoggiBud's container (also available at [4]) and stored in a compressed file generated using *GZip*[1]. This file will ultimately contain distances between all pairs of deliveries.

Afterwards, with an auxiliary heap, the closest m customers for each delivery are mapped and stored so that neighborhoods presented in Section 2.4.1 have a smaller size and can be searched more efficiently. The principle is that far away deliveries tend not to be in the same route. In our experiments, m was set to 100.

2.2 Initial Solution

This topic details the initial solution algorithm, proposing two different greedy procedures. In advance, the second proposed one is more cost-effective when compared to the first, even though it demands more computational resources usage.

The first version (Algorithm 1) does not put effort into creating a feasible solution as it takes into consideration the enhancement algorithms executed afterwards. Even so, after experiments and comparison, the procedure presents a relatively good solution concerning the number of vehicles required.

The algorithms starts by reverse ordering the customers following Equation (1). Then, the resulting array is iterated and a delivery is assigned to a route respecting the following condition: if an existing route exceeds the vehicle capacity, it is skipped. Otherwise, the algorithm inserts the delivery into the route with enough space to serve it. If no route is found, a new one is created.

$$\frac{q}{d_i \times c_{0,i}} \tag{1}$$

- q - Vehicle's capacity;
- d_i - Demand of customer i;
- $c_{0,i}$ - Cost (distance) to go from the depot to customer i.

The second approach consists of the *Clarke-Wright* (CW) constructive heuristic [3], described by Algorithm 2. It starts by generating all possible combination of pairs of deliveries. Then, the resulting combination is ordered respecting its savings, calculated by Equation (2). The algorithm then proceeds to iterating the resulting order merging pairs of sub-routes, consequently resulting in a complete route. If a pair cannot be merged due to one or both of its customers, the algorithm either skips it or inserts the pair into a newly generated empty sub-route . Note that a sub-route is only considered a complete route after the algorithm ends. Therefore, a sub-route is any ordered set of deliveries.

$$c_{0,i} + c_{0,j} - c_{i,j} \tag{2}$$

- $c_{0,i}$ - Distance from the depot to customer i;
- $c_{0,j}$ - Distance from the depot to customer j;
- $c_{i,j}$ - Distance from customer i to customer j.

[1] Available in https://github.com/JuliaIO/GZip.jl

Algorithm 1: *Greedy Initial Solution*

Input: Instance data $\mathcal{P}$
Output: Solution $\mathcal{S}$
greedy($\mathcal{P}$)

```
S ← ∅;  D ← sort(deliveries)
d_t ← 0        // d_t is the current total demand.
for each d ∈ D do
    if demand + d_t ≤ capacity then
        d_t ← d_t + demand
        // insert d in the last position of the
           last route in S
        insert(d, S)
    else
        d_t ← demand
        createRoute(e, S) // create a route for d
         in S
return S
```

Algorithm 2: *Initial Solution Clarke-Wright*

Input: Instance data $\mathcal{P}$
Output: Solution $\mathcal{S}$
ClarkeWright($\mathcal{P}$)

```
S ← ∅
ordered_pairs ← getPairs(deliveries)
foreach (i, j) ∈ ordered_pairs do
    if ∃ route_i or ∃ route_j then
        mergeRoute((i, j), S, P)
    else
        createRoute((i, j), S)
return S
```

Algorithm 3: *mergeRoute*

Input: Delivery pair (i, j); Solution $\mathcal{S}$; Instance data $\mathcal{P}$
mergeRoute((i, j), $\mathcal{S}$, $\mathcal{P}$)

```
if ∃ route_i and ∃ route_j then
    if demand(route_i) + demand(route_j) ≤ capacity
     then
        // concat if i, j are both extreme points in
           different routes
        route_i ← route_i ⊕ route_j
else
    if ∃ route_i then
        // insert delivery j in route_i if i is an
           extreme point and the insertion does not
           exceed capacity
        insert(i, j, route_i)
    else
        insert(j, i, route_j)
```

2.3 Iterated Local Search

Proceeding to the solution enhancement part, it was used the Iterated Local Search (ILS) metaheuristic [5], presented by Algorithm 4. ILS is a technique based on Local Search that has been used successfully to resolve CVRP variants [8, 10]. Being a Local Search based algorithm means that, to improve a current given solution, the procedure evaluates neighborhood moves (modifications in the current solution) and analyze its cost with a delta evaluation strategy.

To briefly describe it, the procedure consists of three parts: an external loop controlling the execution duration, another loop used to disturb the solution and a Random Non-Ascendant (RNA) local search [9].

Initially, the algorithm starts with an initial solution $\mathcal{S}$. Then, it perturbs the input solution by executing random moves belonging to the set of moves $\mathcal{N}$, as an attempt of changing the escaping local optima (lines 4–6). To change the disturbance level, the algorithm uses two counters, $iter$ and p. Once $iter$ equals the procedure's parameter $iter_{max}$, the disturbance level increases. If the perturbation level equals the limit level p_{max}, it is reset to its initial state p_0 (lines 11–14).

The Local Search given by RNA is presented by Algorithm 5. It starts with the perturbed solution as input and once again disturbs it by applying a move. However, the difference is that it only accepts the neighborhood solution if the total distance traveled is less than the previous registered. To avoid an early exit from the search, the counter i, present in the external loop condition (line 3), is reset to zero every time an improving solution is obtained.

Finally, after the RNA execution, ILS might accept its solution or not, depending on how good the solution is (lines 8–10). In case the solution obtained is better than the last best-known solution, the procedure resets both disturbance counters and registers the new solution as the best-known.

Algorithm 4: *Iterated Local Search*

ILS($\mathcal{S}$, p_0, p_{max}, $iter_{max}$, rna_{max})
1 $\mathcal{S}' \leftarrow \mathcal{S}$; $iter \leftarrow 0$; $p \leftarrow p_0$
2 $\mathcal{N} \leftarrow$ neighborhood structures
3 **while** $execution_time \leq limit_execution_time$ **do**
4 **while** $p_i \leq p$ **do**
5 $N \leftarrow$ randomly selected neighborhood from $\mathcal{N}$
6 $\mathcal{S}' \leftarrow$ random neighbor solution in $N(\mathcal{S}')$
7 $\mathcal{S}' \leftarrow$ **RNA**($\mathcal{S}, rna_{max}$)
8 **if** $f(\mathcal{S}') \leq f(\mathcal{S})$ **then**
9 $\mathcal{S} \leftarrow \mathcal{S}'$
10 $iter \leftarrow 0$; $p \leftarrow p_0$
11 **if** $iter \geq iter_{max}$ **then**
12 $p \leftarrow p + p_0$
13 **if** $p > p_{max}$ **then**
14 $p \leftarrow p_0$
15 **return** $\mathcal{S}$

Algorithm 5: *Random non-ascendant method*

Input: Solution $\mathcal{S}$; Limit of iteration with no improvement rna_{max}
RNA($\mathcal{S}$, rna_{max})
1 $i \leftarrow 0$
2 $\mathcal{N} \leftarrow$ neighborhood structures
3 **while** $i \leq rna_{max}$ **do**
4 $i \leftarrow i + 1$
5 $N \leftarrow$ randomly selected neighborhood from $\mathcal{N}$
6 $\mathcal{S}' \leftarrow$ random neighbor solution in $N(\mathcal{S})$
7 **if** $f(\mathcal{S}') \leq f(\mathcal{S})$ **then**
8 **if** $f(\mathcal{S}') < f(\mathcal{S})$ **then**
9 $i \leftarrow 0$
10 $\mathcal{S} \leftarrow \mathcal{S}'$
11 **return** $\mathcal{S}$

2.4 Neighborhoods

Two generic neighborhood structures were developed – N-Insertion (Section 2.4.1) and NM-Swap (Section 2.4.2) – in addition to a post-processing algorithm – Lin-Kernighan (LKH) (Section 2.5) – responsible for intra-route optimization.

2.4.1 N-Insertion. In short, this structure is responsible for selecting a sequence of n customers and reinserting each in the best possible location.

Based on the recreate method proposed by Christiaens and Vanden Berghe [2], the move has one purpose: to reinsert each of the n randomly selected customers in the best routes considering its m closest customers. To do so, there are two possible ways. First, insert each delivery in the selected route, locating it exactly prior to the closest customer selected. The second method iterates the route and analyzes each position and chooses the best one to insert the selected delivery.

Take as an example Figure 1, supposing *Route B* has 2 selected customers to be reinserted, after consulting the previously mentioned structure that stores close customers, it was stated that *Route A* has enough space to store both deliveries. If the first insertion method was the one used, than *Delivery* 1 is the first closest one to *Delivery* 8 and if the second method was chosen, it was defined after iterating the route that the best location for *Delivery* 8 was prior to *Delivery* 1.

2.4.2 NM-Swap. Basically, the swap method exchange sequences between routes. It starts by choosing n and m customers from two different routes and swap places. Note that, if n and m are different there are a mix between swapping and inserting. However, the insertion happens following the sequence, meaning that, the sequence ordering does not change after inserting all customers into one route.

Figure 2 presents an example scenario for the swap move with $n = 2$ and $m = 2$. After randomly selecting a sequence with 2 deliveries in both routes (*Delivery* 1 → *Delivery* 3 in *Route A* and *Delivery* 8 → *Delivery* 10 in *Route B*), its location is mapped and the exchange is executed.

Figure 1: N-insertion example

Figure 2: NM-Swap example

2.5 Lin-Kernighan-Helsgaun

As a post-processing phase, it was used the extension to Lin-Kernighan-Helsgaun (LKH) heuristic, provided by a wrapper [2], to ensure the ordering quality of each route.

In short, the Lin-Kernighan-Helsgaun heuristic is an efficient implementation of the Lin-Kernighan (LK) algorithm [1]. The LK is a generalization of $2-opt$ and $3-opt$ neighborhoods that woks reasonably well for graphs under 50 vertices length. Starting with a feasible solution the heuristic executes exchanges of sub-routes until no more edges exchange results in improvements.

Figure 3 presents an example of the used LKH applied in the resulting routes from Figure 1. In the example, both routes could be optimized by applying the heuristic.

Figure 3: LKH example

2.6 Solution Evaluation

As mentioned previously on Section 1, this section provides the objective function used to evaluate a solution's quality.

As seen in equation 3, the objective function sums the traveling distance for all routes, that is, for all vehicles. In short, this means that the less the better.

$$minimize \sum_{v \in V} \sum_{i,j \in D} d_{i,j} x^{v}_{i,j} \tag{3}$$

- V - Set of vehicles in the solution;

[2] LK wrapper is available in https://github.com/chkwon/LKH.jl

- D - Set of deliveries;
- $d_{i,j}$ - Distance from delivery i to j.
- $x^v_{i,j}$ - Equals 1 if both i and j were assigned to v and 0 otherwise.

It is worth reminding that, in the solution, a route starts and ends at the depot, therefore the depot is considered a delivery in the route.

Besides the objective function, there is one extra constraint used in the proposed technique but not in the evaluation function. In this research, assigning one vehicle to attend to only one customer is not a good strategy, only if no other route supports the customer's demand. Therefore, even though the objective function analysis only the total distance, all algorithms have been implemented in a way to push the number of routes in the solution to the minimum.

3 PRELIMINARY RESULTS

This section presents preliminary results for the proposed methods.

As aforementioned stated in Section 1, all input data was provided by Loggi [4] in JSON format. For the distance calculation, a matrix is stored in a compressed file using distances provided by LoggiBud's container (also available in [4]).

The algorithms proposed were implemented using the programming language ***Julia***[3] version 1.5.3. In total, 33 different instances were executed once with a range of customers from 224 to 7094, and each solution final value was compared to its initial value. We highlight that additional experiments are required. However, since only a subset of the experiments were concluded at the time of writing, only one execution per instance is reported.

The Iterated Local Search algorithm was executed for 900 seconds for each instance in a computer with Intel i7-7500U 2.70Ghz processor and 16GB RAM. After the ILS, a LKH procedure was executed in order to ensure the best route ordering.

Tables 1 and 2 provide information regarding distance and time excution on each of the 33 instances execution. The improvement relation was calculated using the GAP variation from the initial solution to the LKH result, accordingly to Equation (4).

$$\frac{f(LKH) - f(\text{Clarke-Wright})}{f(\text{Clarke-Wright})} \tag{4}$$

- $f(M)$ is the total cost (distance) of the solution obtained by method M.

Table 1 presents the results of our preliminary experiments regarding distance. The table enables concluding that the ILS with post-processing can improve the initial solution cost by approximately 13%. Furthermore, Table 2 compares time execution for all algorithms, also presenting the instance length, that is, the number of deliveries in the instance. As already stated in Section 2.2, both tables provide enough information to conclude that even though the Clarke-Wright algorithm takes longer to execute, it is worthwhile since it results in much better initial solutions than the alternative greedy algorithm.

[3]Julia is available for free in https://julialang.org

Instance	Greedy Result	CW Result	ILS Result	Improvement
1	14878.4243	1807.7638	1565.7721	-0.1339
2	14086.5631	1798.5216	1546.7036	-0.14
3	16961.1421	2100.5703	1760.6698	-0.1618
4	2735.2158	657.7554	589.0094	-0.1045
5	2240.1396	579.5696	504.8587	-0.1289
6	4116.3732	886.7496	755.8084	-0.1477
7	44899.5994	4493.0274	3932.1055	-0.1248
8	46344.6286	4569.6951	4033.0354	-0.1174
9	46926.9661	4633.038	4060.2199	-0.1236
10	58321.0821	5046.4545	4492.6874	-0.1097
11	56513.8337	4785.6918	4301.3503	-0.1012
12	67444.5452	5437.449	4919.4503	-0.0953
13	48251.9974	4424.6111	3894.7895	-0.1197
14	40795.0673	3843.4662	3302.6001	-0.1407
15	57150.31	4972.9447	4330.0029	-0.1293
16	113036.1988	8861.3479	7695.6566	-0.1315
17	114960.6388	9104.769	7887.321	-0.1337
18	112791.4119	8823.9054	7591.9202	-0.1396
19	62288.8182	5375.7169	4719.3052	-0.1221
20	60008.0584	5078.351	4510.2657	-0.1119
21	70322.9291	5746.1211	5051.6542	-0.1209
22	25765.944	2447.5545	2122.6586	-0.1327
23	27865.9274	2612.6394	2265.6842	-0.1328
24	26866.8557	2415.3781	2073.2978	-0.1416
25	28807.4057	2594.1914	2149.7895	-0.1713
26	29402.4101	2601.9472	2215.331	-0.1486
27	28567.0192	2598.2104	2147.7999	-0.1734
28	41375.541	4403.0248	3693.0789	-0.1612
29	40821.3021	4328.5037	3629.6777	-0.1614
30	40526.3298	4224.268	3602.8256	-0.1471
31	40049.5009	3504.34	3083.5071	-0.1201
32	40178.2333	3548.9536	3151.9298	-0.1119
33	40470.6346	3560.5558	3205.0776	-0.0998

Table 1: Preliminary results regarding distance

4 CONCLUSION AND FUTURE WORK

This article presented preliminary results for the Last-Mile Capacitated Vehicle Routing Problem with real-based instances for the cities of *Rio de Janeiro*, *Brasília* and *Belém*. To evaluate the solution, the total distance traveled by all vehicles was considered.

To generate an initial feasible solution we apply a simple greedy algorithm and the Clarke-Wright algorithm. To improve the generated solution an ILS with a post-processing phase and two different neighborhood structures was implemented and compared to both initial values. This approach resulted in an improvement mean of approximately 13% compared to the best initial value in each instance (Clarke-Wright solution).

For future work, we intend to address a dynamic version of the problem and implement different heuristics in order to compare efficiency and solution quality.

Instance	Deliveries	Greedy (s)	CW (s)	ILS (s)
1	1037	0.087	0.653	903.952
2	956	0.086	0.552	903.521
3	1199	0.083	0.767	904.37
4	279	0.071	0.284	901.387
5	224	0.084	0.275	900.699
6	358	0.082	0.311	901.268
7	3728	0.157	6.367	912.603
8	3818	0.117	7.181	912.289
9	3863	0.161	7.399	911.794
10	4954	0.138	11.614	926.778
11	4746	0.135	10.809	925.15
12	5619	0.258	15.818	928.149
13	3959	0.119	6.937	910.004
14	3402	0.145	5.048	908.099
15	4785	0.135	10.684	912.878
16	6835	0.893	26.54	915.609
17	7094	0.359	29.403	915.964
18	6834	0.858	25.925	915.795
19	4445	0.128	9.112	912.186
20	4294	0.181	8.654	912.694
21	5068	0.142	12.299	913.939
22	1947	0.086	1.673	904.959
23	2069	0.114	1.948	905.966
24	1970	0.1	1.801	905.784
25	5391	0.149	12.817	918.5
26	5449	0.149	12.922	920.027
27	5217	0.146	11.33	920.475
28	5190	0.144	12.402	914.433
29	5164	0.144	12.206	914.745
30	5068	0.138	11.925	914.219
31	3742	0.16	5.406	918.042
32	3726	0.113	5.286	919.041
33	3759	0.113	5.628	921.474

Table 2: Preliminary results regarding time execution

REFERENCES

[1] Marília M. Caiano. 2018. *Aplicação do Problema do Caixeiro Viajante numa empresa de distribuição: A Heurística de Lin Kernighan.* Ph. D. Dissertation. Universidade de Coimbra.

[2] Jan Christiaens and Greet Vanden Berghe. 2020. Slack Induction by String Removals for Vehicle Routing Problems. *Transportation Science* 54, 2 (2020), 417–433. $$Uhttps://lirias.kuleuven.be/retrieve/545845$$DSISRs_2018.pdf[freelyavailable]

[3] Richard C. Larson and Amedeo R. Odoni. [n. d.]. *Urban Operations Research.* Prentice-Hall. https://web.mit.edu/urban_or_book/www/book/chapter6/6.4.12.html

[4] Loggi. 2021. loggiBUD: Loggi Benchmark for Urban Deliveries. https://github.com/loggi/loggibud.

[5] Helena R. Lourenço, Olivier C. Martin, and Thomas Stützle. 2010. *Iterated Local Search: Framework and Applications.* Springer US, Boston, MA, 363–397. https://doi.org/10.1007/978-1-4419-1665-5_12

[6] Jorge Oyola, Halvard Arntzen, and David L. Woodruff. 2017. The stochastic vehicle routing problem, a literature review, Part II: solution methods. *EURO Journal on Transportation and Logistics* 6, 4 (2017), 349–388. https://doi.org/10.1007/s13676-016-0099-7

[7] Jorge Oyola, Halvard Arntzen, and David L. Woodruff. 2018. The stochastic vehicle routing problem, a literature review, part I: models. *EURO Journal on Transportation and Logistics* 7, 3 (2018), 193–221. https://doi.org/10.1007/s13676-016-0100-5

[8] Puca H. V. Penna, Anand Subramanian, Luiz S. Ochi, Thibaut Vidal, and Christian Prins. 2019. A hybrid heuristic for a broad class of vehicle routing problems with heterogeneous fleet. *Annals of Operations Research* 273 (2019), 5–74. https://doi.org/10.1007/s10479-017-2642-9

[9] Haroldo G. Santos, Túlio A. M. Toffolo, Cristiano LTF Silva, and Greet Vanden Berghe. 2019. Analysis of stochastic local search methods for the unrelated parallel machine scheduling problem. *International Transactions in Operational Research* 26, 2 (2019), 707–724.

[10] Anand Subramanian, Eduardo Uchoa, and Luiz S. Ochi. 2013. A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research* 40, 10 (2013), 2519–2531. https://doi.org/10.1016/j.cor.2013.01.013

[11] Túlio A. M. Toffolo, Thibaut Vidal, and Tony Wauters. 2019. Heuristics for vehicle routing problems: Sequence or set optimization? *Computers & Operations Research* 105 (2019), 118–131. https://doi.org/10.1016/j.cor.2018.12.023

[12] Eduardo Uchoa, Diego Pecin, Artur Pessoa, Marcus Poggi, Thibaut Vidal, and Anand Subramanian. 2017. New benchmark instances for the Capacitated Vehicle Routing Problem. *European Journal of Operational Research* 257, 3 (2017), 845–858. https://doi.org/10.1016/j.ejor.2016.08.012

Busca Tabu aplicada a resolução de uma variante do problema roteamento dinâmico de veículos

Miguel Rodrigues Guimarães de Oliveira
miguel.oliveira2@estudante.ufla.br
DCC, Universidade Federal de Lavras
Lavras, Minas Gerais, Brazil

Arthur Henrique Sousa Cruz
arthur.cruz1@estudante.ufla.br
DCC, Universidade Federal de Lavras
Lavras, Minas Gerais, Brazil

Franklina Maria Bragion Toledo
fran@icmc.usp.br
ICMC, Universidade de São Paulo
São Carlos, São Paulo, Brazil

Mayron César O. Moreira
mayron.moreira@ufla.br
DCC, Universidade Federal de Lavras
Lavras, Minas Gerais, Brazil

ABSTRACT

Companies that work with logistical planning have invested in technological and labor resources to develop routing algorithms. This effort is due, particularly, to the logistical demand in deliveries, accelerated in 2020. Motivated by this context, this study addresses the *dynamic pickup and delivery problem with time windows* (DPDPTW). Vehicles are uncapacitated and must attend initial requests and new ones that are dynamically generated.

The generation of new requests does not follow any defined probability distribution. The optimization criteria are lexicographically defined by: (i) maximizing the number of requests served and (ii) minimizing the cost of routes. Reoptimizations are carried out in a planning horizon operationally divided into time intervals. Thus, requests arriving in a period j are considered at the beginning of $j + 1$.

DPDPTW is solved using the Tabu Search algorithm. This metaheuristic is often adapted to vehicle routing problems in the literature. For this purpose, three operators for inserting new requests and one operator for generating the neighborhood are considered. Computational tests with two instances with a thousand points, inspired by practical contexts, show the efficiency of the Tabu Search and the "Non-fixed Location Insertion Heuristic" procedure in dynamically generated requests.

RESUMO

Empresas que trabalham com o planejamento logístico têm investido recursos tecnológicos e de mão-de-obra para o desenvolvimento de algoritmos de roteamento. Esse empenho se deve, em especial, devido à demanda logística na entrega de pedidos que acelerou em 2020. Motivado por esse contexto, este estudo aborda o *problema de coleta e entrega dinâmica com janelas de tempo* (PCEDJT). Os veículos são não capacitados e devem atender os pedidos iniciais e os novos pedidos que dinamicamente são gerados.

A geração dos novos pedidos não segue nenhuma distribuição de probabilidade definida *a priori*. A otimização é guiada por duas funções, que lexicograficamente seguem a prioridade: (i) maximização do número de pedidos atendidos, e (ii) minimização do custo das rotas. As reotimizações dos pedidos são feitas em um horizonte de planejamento dividido operacionalmente em intervalos de tempo. Assim, os pedidos que chegam em um período j são considerados no início do período $j + 1$.

A resolução do PCEDJT se dá a partir do Algoritmo Busca Tabu, uma metaheurística frequentemente adaptada para problemas de roteamento de veículos na literatura. Para tanto, considera-se três operadores de inserção de novos pedidos e um operador para geração da vizinhança. Através de testes computacionais com duas instâncias com mil pontos, inspiradas em contextos práticos, comprova-se a eficiência da Busca Tabu e do procedimento de "Inserção de pontos não-fixos" em pedidos gerados dinamicamente.

CCS CONCEPTS

• **Applied computing** → **Transportation**.

KEYWORDS

dynamic vehicle routing problems, heuristics, tabu search, combinatorial optimization

PALAVRAS-CHAVE

roteamento dinâmico de veículos, heurísticas, busca tabu, otimização combinatória

ACM Reference Format:
Miguel Rodrigues Guimarães de Oliveira, Arthur Henrique Sousa Cruz, Franklina Maria Bragion Toledo, and Mayron César O. Moreira. 2021. Busca Tabu aplicada a resolução de uma variante do problema roteamento dinâmico de veículos. In *Workshop PBP-Loggi Resultados do Programa de Bolsas de Pesquisa (PBP-Loggi), 08 de Dezembro, 2021, Evento Virtual.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUÇÃO

A demanda logística na entrega de pedidos acelerou em 2020, e estudos indicam que haverá uma intensificação no volume serviços de entrega para os próximos três anos [5]. Diante disso, empresas que trabalham com o planejamento logístico têm investido recursos tecnológicos e de mão-de-obra para o desenvolvimento de algoritmos de roteamento. O objetivo consiste em otimizar os percursos

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Workshop PBP-Loggi, 08 de Dezembro, 2021, Evento Virtual
© 2021 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/1122445.1122456

de seus veículos, reduzindo o tempo ou o custo das rotas, como também maximizando o número de clientes atendidos.

A roteirização de pedidos pode ser feita de maneira estática, na qual as requisições são conhecidas no início do horizonte de planejamento e não há novos pontos a serem visitados por veículos na etapa operacional. Por outro lado, devido ao comércio eletrônico, cada vez mais os sistemas computacionais de apoio à decisão operacionalizam pedidos de maneira dinâmica. Logo, o conjunto de demandas de entrega pode aumentar ao longo do período. Isso dificulta a adaptação de rotas já obtidas considerando os pedidos anteriores, arranjados de forma estática [8]. Esse cenário levou ao surgimento de uma nova variação do PRV, o Problema de Roteamento Dinâmico de Veículos (PRDV), que lida com a adição de novos pedidos dinamicamente ao longo de uma jornada de trabalho [13, 14].

Ser capaz de obter boas rotas, apesar da dinamicidade do cenário, resulta em uma vantagem competitiva importante nesse mercado, já que o planejamento de rotas possui influência nos custos de entrega [7, 19]. Existem trabalhos na literatura que estudam uma forma eficiente para o roteamento de veículos no contexto de coletas e entregas dinâmicas [3, 6, 8, 15]. No entanto, [14] indicam que existem poucas abordagens na literatura que são intrinsecamente dinâmicas. Para os autores, a maioria dos estudos adaptam abordagens estáticas para lidar com a característica dinâmica do PRDV. Apesar de incentivarem o estudo de metodologias dinâmicas em sua essência, os autores afirmam que métodos adaptados obtém resultados satisfatórios em instâncias da literatura.

Mais especificamente para o caso das empresas de entrega de pacotes, torna-se necessária a coleta e a entrega de cada pedido, sendo que ambos os serviços devem ser realizados dentro de janelas de tempo de atendimento específicas. Essas janelas são limitantes para o horário de recolhimento ou de entrega dos pedidos [16]. Além disso, uma grande quantidade de pedidos são adicionados ao longo do dia, o que atribui dinamicidade ao problema [6, 8]. Essas são as principais características do problema estudado neste projeto, o qual é denominado Problema de Coleta e Entrega Dinâmica com Janelas de Tempo Dinâmico (PCEDJT).

Este trabalho contribui com o estudo do PCEDJT em quatro vertentes:

(1) Implementação de uma adaptação do algoritmo Busca Tabu, inspirada no trabalho de [10].
(2) Validação das soluções da Busca Tabu através de experimentos com instâncias reais, advindas do *framework Loggi-BUD* [11] e do trabalho de [18].
(3) Avaliação de heurísticas de inserção e do potencial de melhoria da Busca Tabu em relação às soluções obtidas por uma heurística construtiva.
(4) Disponibilizar, de forma livre e aberta, uma implementação para a resolução do PCEDJT. Isso auxilia no processo de comparação de soluções com outros métodos, a serem propostos por outros pesquisadores que desejam avançar como o estado da arte do problema em questão.

O restante desse artigo está organizado da seguinte forma. A Seção 2 apresenta a descrição formal do PCEDJT. Na Seção 3, apresenta-se as etapas da Busca Tabu implementada neste trabalho, inspirada no trabalho de [10]. A Seção 4 reporta um conjunto de experimentos computacionais para avaliação do desempenho da Busca Tabu através de instâncias advindas de contextos reais. Conclusões e perspectivas de trabalhos futuros são discutidas na Seção 5.

2 DESCRIÇÃO DO PROBLEMA

O Problema de Coleta e Entrega Dinâmica com Janelas de Tempo (PCEDJT) consiste em planejar as rotas para coleta e entrega de encomendas. O atendimento de cada pedido deve respeitar janelas de tempo pré-estabelecidas, dentro de uma jornada de trabalho fixa. Os objetivos são definidos hierarquicamente da seguinte forma: (i) maximizar a quantidade de pedidos atendidos, e (ii) minimizar a distância percorrida pelos veículos. Observa-se que no trabalho de [10], base para o método de solução proposto neste estudo, a função objetivo consiste somente na minimização da distância total de todas as rotas (item (ii) citado anteriormente).

Sejam $P = \{1, 2, ..., n\}$ e $D = \{n+1, n+2, ..., 2n\}$ os conjuntos de pontos de coleta e entrega, respectivamente, e seja $V = P \cup D$. O conjunto de pedidos R é composto por tuplas tal que $\forall r \in R$, $r = (i, i+n)$, com $i \in P$ e $i+n \in D$. Para cada $v \in V$, define-se s_v como o tempo necessário para se realizar o serviço (de coleta ou entrega) em v. Este serviço deve ser iniciado dentro de uma janela de tempo dada pelo intervalo $[e_v, l_v]$.

Considere uma frota heterogênea F de veículos não capacitados candidatos a serem utilizados, de forma que cada veículo é associado a um único motorista e atende uma única rota. Como se trata de um ambiente dinâmico, sua rota pode sofrer modificações durante o dia. Cada veículo utilizado parte e retorna de um mesmo depósito, denotado pelo ponto o. Os veículos possuem as mesmas janelas de tempo, definidas por $[\underline{h}, \overline{h}]$. Esse intervalo contempla a jornada de trabalho estabelecida (h), ou seja, o horizonte de planejamento.

Tem-se o conjunto de períodos $\Omega = \{\Omega_1, \Omega_2, ..., \Omega_{|\Omega|}\}$, com $\Omega_j = [p_j, p_{j+1})$, $j = 1, ..., \mid \Omega \mid$, em que p_j e p_{j+1} correspondem ao tempo de início e fim do período j. Considera-se $p_1 = \underline{h}$ e $p_{\Omega_{|\Omega|}+1} = \overline{h}$. Portanto, $\bigcup_{j=1}^{|\Omega|} \Omega_j$ representa a jornada de trabalho $h = [\underline{h}, \overline{h}]$ de um dia de um motorista, e Ω_0 um intervalo anterior ao início da jornada. A partir do início de cada período Ω_j da jornada de trabalho, um novo conjunto de pedidos (dinâmicos) é recebido (Q_{Ω_j}, $0 \leq j <\mid \Omega \mid$). Nota-se que pedidos que chegam em $\Omega_{|\Omega|}$ não serão considerados para a solução. Inicialmente, tem-se um conjunto de pedidos $Q_{\Omega_0} = R$, conhecido antes do início do horizonte de planejamento, que corresponde ao conjunto de pedidos estáticos. Portanto, que o conjunto de todos os pedidos é dado por $\mathcal{R} = \bigcup_{j=0}^{|\Omega|-1} Q_{\Omega_j}$. Neste estudo, considera-se que o surgimento dos novos pedidos não tenha alguma distribuição de probabilidade definida *a priori*.

Uma solução para o PCEDJT consiste em um conjunto de rotas que deve respeitar as restrições apresentadas a seguir.

- Precedências de coleta e entrega de cada pedido: para um pedido $r = (i, i+n)$, o ponto de entrega $(i+n) \in D$ não pode ser visitado antes de seu correspondente ponto de coleta $i \in P$.

- Janelas de tempo dos pontos em V e dos veículos utilizados: cada veículo $f \in F$, que for utilizado, deve partir e retornar do depósito no intervalo $[\underline{h}, \overline{h}]$. Além disso, a chegada do veículo f ao ponto v não pode exceder l_v. Caso o motorista do veículo f chegue antes de e_v, o mesmo deve esperar até que o tempo atual do percurso alcance e_v.
- Turno de trabalho: a quantidade de horas trabalhadas por motorista não pode exceder h unidades de tempo.
- Exclusividade de visita: em um pedido $r = (i, i+n)$, cada ponto $i \in V$ e $i+n \in V$ pode ser visitado exatamente uma vez, por um mesmo veículo.
- Obrigatoriedade de visita: em um pedido $r = (i, i+n)$, a visita ao ponto $i \in N$ por um veículo $f \in F$ implica na visita ao ponto $i+n \in N$ pelo mesmo veículo.

O conjunto de rotas (ou solução) do período Ω_j, denotado por $\mathcal{S}_{\Omega_j}$, considera como elegíveis os pedidos $Q_{\Omega_{j-1}}$ e os pedidos de todos os períodos anteriores Q_{Ω_k}, $\Omega_k < \Omega_j$, ainda não atendidos (toma-se $k \geq 0$ e $j > 0$). Se o pedido $r = (i, i+n) \in \mathcal{R}$, que possui ponto de coleta i presente na solução $\mathcal{S}_{\Omega_{j-1}}$ $(j = 2, ..., \mid \Omega \mid)$ for iniciado, seu par de entrega $(i+n)$ deverá estar presente em $\mathcal{S}_{\Omega_{k'}}$, com $\Omega_j \leq \Omega_{k'} \leq \Omega_{|\Omega|}$, caso o serviço $i+n$ não tenha sido finalizado até o período Ω_{j-1}.

Define-se o grafo ponderado $G = (N, A)$, com $N = V \cup \{o\}$ e $A = N \times N$, tal que $\forall (u, v) \in A$, tem-se um custo d_{uv} e um tempo de viagem t^f_{uv}, $f \in F$, $u \neq v$. Para a definição dos objetivos a serem otimizados, consideramos as variáveis:

- $y_r \in \{0, 1\}$: variável binária igual a 1 se o pedido $r \in \mathcal{R}$ foi atendido;
- $x_{uvf} \in \{0, 1\}$: variável binária igual a 1 se o veículo $f \in F$ passa pelo trecho que liga u a v.

Os objetivos a serem otimizados de forma lexicográfica são apresentados abaixo:

Objetivo 1

$$\max \sum_{r \in \mathcal{R}} y_r \tag{1}$$

Objetivo 2

$$\min \sum_{f \in F} \sum_{(u,v) \in A} d_{uv} x_{uvf} \tag{2}$$

A escolha das funções descritas pelo "Objetivo 1" e "Objetivo 2" são justificadas pelo fato desta pesquisa fazer parte de um projeto de mestrado, que adota os mesmos critérios de qualidade. Além disso, dado um período $\Omega_j = [p_j, p_{j+1})$, ressalta-se que o mecanismo de reotimização abordado aqui não altera a chegada ao vértice $v \in V$ se um motorista já estiver em curso para sua visita. Além disso, as requisições dinâmicas geradas no período Ω_j serão processadas (não necessariamente alocadas) no instante p_{j+1}. Essa estratégia difere de [10], que propõe uma reinserção de pedidos contínua a medida que eles são criados.

3 BUSCA TABU

A metaheurística Busca Tabu foi introduzida na literatura por [9]. A ideia do método é encontrar soluções de qualidade, quanto ao objetivo a ser otimizado, através da geração de um conjunto de soluções obtido por meio de operadores de vizinhança. O método busca escapar de mínimos locais permitindo que uma solução incumbente piore sua função objetivo, caso não haja alguma solução vizinha melhor. Para evitar ciclagem, a Busca Tabu proíbe que alguns movimentos sejam revertidos por um determinado número de iterações.

Este estudo apresenta uma implementação do algoritmo Busca Tabu, baseada no trabalho de [10]. A escolha por tal algoritmo se dá devido à sua eficiência comparado com o método de [12], apresentando melhorias para maior parte das instâncias artificiais do PCEDJT, incluindo problemas de 1.000 pedidos. Divide-se a explicação da metaheurística da seguinte forma: primeiramente, o pseudocódigo da Busca Tabu (Seção 3.1) implementada é apresentado. Na sequência, as Seções 3.2, 3.3 e 3.4 especificam como foram desenvolvidos os procedimentos de solução inicial, os operadores vizinhança e o gerenciamento da lista tabu, respectivamente.

3.1 Pseudocódigo

O Algoritmo 1 apresenta um esquema genérico referente às etapas da Busca Tabu. O arcabouço delineado por [9] prevê a aplicação de um procedimento denominado *critério de aspiração*, permitindo que um movimento proibitivo seja aceito se o mesmo produzir um aprimoramento na melhor solução obtida até o momento. Todavia, [10] mostram experimentalmente que para o PCEDJT, o impacto do *critério de aspiração* no resultado final da metaheurística era marginal, mesmo diante de instâncias com 1.000 pontos. Assim, decidiu-se por não testar essa estratégia neste estudo.

A Linha 1 indica a geração de uma solução factível inicial para o problema. Na implementação desenvolvida, considera-se uma heurística construtiva de inserção. A partir de uma sequência pré-determinada de requisições e de um conjunto de rotas, para cada requisição, tenta-se a inserção de seus pontos de coleta e entrega em posições de uma rota, que minimize o custo adicional da solução parcial. As inserções devem garantir a factibilidade da solução, com base nas restrições descritas na Seção 2. Uma explicação detalhada desse algoritmo pode ser vista na Seção 3.2.

Na Linha 2, configura-se que, inicialmente, a variável que armazena a melhor solução a ser retornada pela Busca Tabu (denotada por $\overline{s}$) seja igual à solução inicial. O laço de repetição principal do algoritmo está relacionado às Linhas 3-12. Dois critérios foram implementados como condições de parada: (i) se não houve ao menos uma solução vizinha factível, e se (ii) o número de iterações sem melhoria da melhor solução encontrada pela Busca Tabu, até o momento, ultrapassou $|R|$ iterações. Ambos os critérios devem ser obedecidos para não interromper o funcionamento da Busca Tabu.

As Linhas 5-9 exploram a vizinhança da solução incumbente s, denotada por $\mathcal{N}(s)$. Este estudo considera a implementação dos operadores de vizinhança: *inserção simples* (IS), *inserção aleatória* (IA), *inserção de requisições não-fixas* (IR) e *inserção de pontos não-fixos* (IP). A Seção 3.3 mostra como cada um desses algoritmos funciona. Nota-se que o algoritmo considera apenas vizinhos de s gerados por movimentos não-tabu. Existe a possibilidade da Busca Tabu encontrar soluções piores que s, o que traz como vantagem uma maior diversidade no processo exploratório da busca. Na Linha 13, o algoritmo finaliza sua execução retornando a melhor solução encontrada.

1 Seja s uma solução factível, obtida a partir de uma heurística construtiva ou uma heurística de inserção (abordagem dinâmica);
2 $\bar{s} \leftarrow s$; ▷$\bar{s}$: solução retornada pelo algoritmo
3 **while** *critério de parada não foi atingido* **do**
4 Seja b a solução que armazena o melhor vizinho de s; ▷inicialmente, $b \rightarrow \emptyset, f(b) = \infty$
5 **for** $s' \in \mathcal{N}(s)$ **do**
6 **if** $f(s') < f(b)$ *e* s' *não é tabu* **then**
7 $b \leftarrow s'$;
8 **end**
9 **end**
10 $s \leftarrow b$;
11 Atribua como tabu o(s) movimento(s) que geraram a solução b;
12 **end**
13 **return** $\bar{b}$; ▷melhor solução encontrada pela Busca Tabu

Algoritmo 1: Busca Tabu

3.2 Solução inicial

Os Algoritmos 2 e 3 apresentam a heurística construtiva para a geração da solução inicial da Busca Tabu (Linha 1, Algoritmo 1). Considerando o Pseudocódigo 3, nas Linhas 1-3, as variáveis e conjuntos auxiliares para o planejamento das rotas são configurados. O laço repetitivo correspondente às Linhas 4-12 indica que a heurística finalizará somente quando cada pedido for alocado a alguma rota. Adota-se, nesta implementação, um sequenciamento de pedidos determinado da seguinte forma: os pedidos são divididos em 10 grupos de 20 a 40 nós, cada. Com isso, a cada inserção, um grupo pacotes novo é alocado na rota da solução final.

A rota mais apropriada, assim como as posições de inserção dos pontos de coleta e entrega do pedido r, são obtidos na Linha 8, através da chamada do Algoritmo 2. Este procedimento, por sua vez, utiliza como critério de qualidade o posicionamento dos pontos referentes ao pedido r em uma rota do conjunto $\tilde{\mathcal{R}}$, que minimize o custo adicional da solução (Linhas 3-13, Algoritmo 2). Por fim, a heurística construtiva retorna a solução factível s (Linha 13, Algoritmo 3).

Data: $r = (p, d)$: pedido r, com seus respectivos pontos de coleta p e entrega d; $\tilde{\mathcal{R}}$: conjunto de rotas.
1 Seja $\beta \leftarrow \infty$ o valor do menor aumento de custo na função objetivo; ▷O custo se refere ao Objetivo 2, Seção 2.
2 Sejam u_b, p_b e d_b a melhor rota, a melhor posição de coleta em u_b e a melhor posição de entrega em u_b que minimizam o aumento do custo da solução, respectivamente;
3 **for** $u \in \tilde{\mathcal{R}}$ **do**
4 **for** *Todas as posições factíveis de* p *em* u **do**
5 Insira p em u;
6 **for** *Todas as posições factíveis de* d *em* u **do**
7 Insira d em u;
8 Seja δ o aumento do custo das rotas ao inserir $r = (p, d)$ em u;
9 **if** $\delta < \beta$ **then**
10 $\beta \leftarrow \delta, u_b \leftarrow u, p_b \leftarrow p, d_b \leftarrow d$;
11 **end**
12 **end**
13 **end**
14 **end**
15 **return** u_b, p_b, d_b;

Algoritmo 2: Algoritmo de Melhor Inserção - MelhorIns.

1 Seja m o número de veículos usados; ▷$m \leftarrow 0$.
2 Seja s uma solução parcial; ▷$s \leftarrow \emptyset$.
3 Seja X o conjunto de pedidos estáticos iniciais; ▷$X \leftarrow R$.
4 **while** $X \neq \emptyset$ **do**
5 Seja r um pedido qualquer em X;
6 $X = X \backslash \{r\}$;
7 Seja Y o conjunto de rotas em s;
8 $u_b, p_b, d_b \leftarrow$ MelhorIns(r, Y);
9 **if** u_b *é uma rota nova em* s **then**
10 $m \leftarrow m + 1$;
11 **end**
12 **end**
13 **return** s; ▷solução factível do PCEDJT

Algoritmo 3: Heurística construtiva.

3.3 Operadores de Vizinhança

A presença dos operadores de vizinhança é notada na Linha 5 do Algoritmo 1. A cada período de reotimização, a Busca Tabu é executada, considerando os pedidos ainda não atendidos e os novos pedidos, que surgiram de forma dinâmica. A ordem com que a inserção dos pedidos é feita obedece a sequência de surgimento dos pedidos dinâmicos. Um total de três operadores foram implementados, como descrito a seguir:

Inserção Simples (IS): Trata todos os pedidos já presentes em uma rota como fixos. Na sequência, encontra a posição do pedido r na posição não-tabu de uma rota, que minimiza o custo adicional.

Inserção de Requisições não-fixas (IR): Considerando fixos os pedidos em o ponto de coleta já foi visitado, esse operador busca inserir os novos pedidos e os remanescentes da solução na rota e na posição mais apropriada, que minimize o custo adicional na solução.

Inserção de pontos não-fixos (IP): Considerando fixos os pontos (coleta ou entrega) que já foram visitados, esse operador busca inserir os novos pedidos e os remanescentes da solução na rota e na posição mais apropriada prezando sempre pela, que minimize o custo adicional na solução.

A função objetivo principal do PCEDJT é a maximização do número de pedidos atendidos. Assim, os três operadores de vizinhança da Busca Tabu tentam inserir o maior número possível de pedidos. Em cada execução do Algoritmo 1, apenas o operador IS é executado como gerador de soluções vizinhas. No entanto, para a inserção dos novos pedidos a cada período, os três operadores

de inserção são testados, um a cada execução da metaheurística, gerando um total de três variações da Busca Tabu.

3.4 Lista Tabu

A Lista Tabu implementada neste trabalho consiste em uma fila, de tamanho $|R|$ (quantidade de pedidos que podem ser removidos), com elementos de três coordenadas correspondentes a rota, o identificador do nó e a sua posição na rota. Essa tripla de valores configura os movimentos tabu. Caso a fila atinja sua capacidade máxima, retira-se o primeiro elemento da mesma, e insere-se o novo movimento tabu na última posição da estrutura.

4 RESULTADOS COMPUTACIONAIS

Os experimentos computacionais foram feitos em um computador com 8GB de RAM, processador Intel Core i5-7200U CPU @ 2.50GHz e 64 bits, memória principal ATA Disk de 1TB, com o sistema operacional Ubuntu 18.04.6 LTS. A implementação foi feita na linguagem Python 3.9. O código, as instâncias utilizadas e os resultados computacionais estão disponíveis em https://github.com/ElMigu17/iniciacao_cientifica.

Duas instâncias foram utilizadas como base dos testes: uma advinda de [18] e a outra gerada pelo LoggiBUD[11], ambas com mil nós. A primeira foi ramificada em duas variações: a versão original, extraída de [18] (SB1), e uma variante com 30% dos nós escolhidos aleatoriamente para fazerem par com o depósito. Características adicionais referentes as instâncias são descritas abaixo.

- Instâncias "SB1" e "SB2" [18]: têm horizonte de tempo igual a 480, tamanho de janelas de tempo com valor de 60, com o início e fim sempre dentro do horizonte de tempo. A capacidade de todos os veículos igual a 300, e tamanho dos pacotes variam de 10 a 190. As distâncias são iguais aos tempos e aos custos, os quais foram geradas usando as ferramentas Open Source Routing Machine (OSMR) a partir de dados de geolocalização do Open Source Vehicle Routing Instance Generator (OVIG).
- Instância "Loggi" [11]: A capacidade de todos os veículos igual a 180, tamanho dos pacotes variando de 1 a 10. As distâncias são iguais aos tempos e aos custos, geradas usando o OSMR a partir de dados de geolocalização do LoggiBUD.

Três rodadas de testes foram executados, alterando em cada uma qual heurística de inserção dinâmica seria usada: simples (operador IS), com requisições não-fixas (operador IR), e com pontos não-fixos(operador IP). Como preparo das instâncias, cada uma das três foi dividida em 10 partes com número de pedidos semelhantes. A justificativa dessa metodologia se dá para simular a chegada de novos pedidos após um determinado tempo. Dessa forma, a primeira parte é inserida estaticamente e as restantes são inseridas dinamicamente, com intervalos de tempo iguais entre as inserções.

Na Tabela 1, são reportados os resultados obtidos para as instâncias consideradas. Na primeira coluna (denominada "Instância"), é descrito o nome da instância. A Coluna 2 indica a heurística de inserção utilizada para a etapa dinâmica (Ins). Em todos os testes, os algoritmos conseguiram atender todos os pedidos. Assim, na Coluna 3, apresenta-se a média do percentual de melhoria obtida pela Busca Tabu (BT) em relação a solução inicialmente obtida pela heurística construtiva de inserção (HC) (de nome M, definida por $M = ((HC - BT)/HC) \times 100$), com relação ao **Objetivo 2** (Seção 2). O tempo computacional (em segundos) de cada heurística de inserção (quando os pedidos dinâmicos são gerados) separadamente, e da Busca Tabu (sem a etapa construtiva e a de inserção) são apresentados nas Colunas 4 e 5, respectivamente.

Table 1: Resultados computacionais.

Instância	Ins	M (%)	TI (s)	TT (s)
SB1	IS	2,5	0,2	4,7
SB1	IR	4,0	0,5	22,6
SB1	IP	7,1	1,0	22,1
SB2	IS	1,5	0,1	2,1
SB2	IR	4,8	0,4	8,9
SB2	IP	8,8	0,7	9,4
Loggi	IS	9,5	0,4	8,9
Loggi	IR	2,7	0,7	15,6
Loggi	IP	8,0	87,2	24,0

Considerando a Heurística Construtiva (Algortimos 2 e 3) e a Busca Tabu (Algoritmo 1), nota-se que a Busca Tabu conseguiu aprimorar o custo total nos nove testes. A regra de inserção IP foi a que obteve o melhor percentual de aprimoramento do **Objetivo 2**, seguido pelas regras IS e IR (a exceção da instância Loggi, cujo operador IS foi melhor que IS). O desempenho satisfatório da inserção de pontos não-fixos (IS) pode ser explicado pela maior flexibilidade que essa heurística provê em relação às demais. Com relação ao tempo computacional reportado, observa-se uma discrepância de eficiência em todos os cenários, quando os operadores IR e IP são selecionados. O motivo se deve à qualidade da solução inicial provida por IR e IP à Busca Tabu, que posterga a convergência da metaheurística, levando a uma solução de menor valor de função objetivo.

5 CONCLUSÕES E TRABALHOS FUTUROS

Este estudo aborda o *problema de coleta e entrega dinâmica com janelas de tempo* (PCEDJT). Os veículos são não capacitados e devem atender os pedidos iniciais e os novos pedidos que dinamicamente são gerados. A geração dos novos pedidos não segue nenhuma distribuição de probabilidade definida *a priori*. A otimização é guiada por duas funções, que lexicograficamente seguem a prioridade: (i) maximização do número de pedidos atendidos, e (ii) minimização do custo das rotas. As reotimizações dos pedidos são feitas em um horizonte de planejamento dividido operacionalmente em intervalos de tempo. Assim, os pedidos que chegam em um período j são considerados no início do período $j + 1$.

A resolução do PCEDJT se deu a partir do Algoritmo de Busca Tabu descrito na Seção 3, motivado pela relevância dos resultados apresentados por [10]. O algoritmo possui um esquema clássico, com uma solução construtiva inicial randômica, e um operador de vizinhança que insere novos pedidos e reotimiza as rotas já pré-existentes na solução. Testes com duas instâncias adaptadas a partir de dados reais mostraram que existe melhora significativa ao se usar a Busca Tabu em conjunto com qualquer heurística de inserção. Com relação às heurísticas de inserção, os melhores procedimentos foram, nessa ordem: Inserção de pontos não-fixos (IP), Inserção de Requisições não-fixas (IR) e Inserção Simples (IS).

Trabalhos futuros consideram as seguintes possibilidades: (i) utilizar as instâncias disponíveis no LoggiBUD; (ii) desenvolvimento de testes experimentais com mais instâncias inspiradas em contextos reais; (iii) estudo do impacto do critério de aspiração na Busca Tabu, por meio de testes com novas instâncias; (iv) calibração dos parâmetros da Busca Tabu através de pacotes tais como HyperOpt [4] ou Optuna [1]; (v) utilização de heurísticas de decomposição para a redução do tamanho de instâncias de grande porte [17]; e (vi) aplicação de algoritmos de aprendizado estatístico para a identificação de padrões durante a busca, reduzindo assim o esforço computacional de metaheurísticas [2].

AGRADECIMENTOS

Este trabalho foi realizado com o apoio da Loggi (www.loggi.com), por meio do Programa de Bolsas de Pesquisa Loggi (PBP-LOGGI). Os autores ainda agradecem a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e a Fundação de Desenvolvimento Científico e Cultural (FUNDECC/UFLA, Contrato 032/2020).

REFERENCES

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[2] Florian Arnold, Ítalo Santana, Kenneth Sörensen, and Thibaut Vidal. 2021. PILS: Exploring high-order neighborhoods by pattern mining and injection. *Pattern Recognition* 116 (2021), 107957.

[3] Matthias Bender, Jörg Kalcsics, and Anne Meyer. 2020. Districting for Parcel Delivery Services–A Two-Stage Solution Approach and a Real-World Case Study. *Omega* (2020), 102283.

[4] James Bergstra, Daniel Yamins, and David Cox. 2013. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning*. PMLR, 115–123.

[5] Reuters Events. 2021. E-commerce growth contributing to an increasing global demand for logistics space. https://www.reutersevents.com/supplychain/ecommerceretail/e-commerce-growth-contributing-increasing-global-demand-logistics-space/. [Online; acessado em 25 de novembro de 2021].

[6] Michel Gendreau, François Guertin, Jean Yves Potvin, and René Séguin. 2006. Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries. *Transportation Research Part C: Emerging Technologies* 14, 3 (2006), 157–174. https://doi.org/10.1016/j.trc.2006.03.002

[7] Michel Gendreau, François Guertin, Jean Yves Potvin, and Éric Taillard. 1999. Parallel tabu search for real-time vehicle routing and dispatching. *Transportation Science* 33, 4 (1999), 381–390. https://doi.org/10.1287/trsc.33.4.381

[8] Gianpaolo Ghiani, Emanuele Manni, Antonella Quaranta, and Chefi Triki. 2009. Anticipatory algorithms for same-day courier dispatching. *Transportation Research Part E: Logistics and Transportation Review* 45, 1 (2009), 96–106. https://doi.org/10.1016/j.tre.2008.08.003

[9] Fred Glover and Manuel Laguna. 1998. Tabu search. In *Handbook of combinatorial optimization*. Springer, 2093–2229.

[10] Penny Louise Holborn. 2013. Heuristics for Dynamic Vehicle Routing Problems with Pickups and Deliveries and Time Windows A thesis. May (2013), 252. http://orca.cf.ac.uk/47742/8/2013holbornplphdV.3.pdf

[11] Loggi. 2021. loggiBUD: Loggi Benchmark for Urban Deliveries. https://github.com/loggi/loggibud.

[12] Snežana Mitrović-Minić and Gilbert Laporte. 2004. Waiting strategies for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological* 38, 7 (2004), 635–655. https://doi.org/10.1016/j.trb.2003.09.002

[13] Victor Pillac, Michel Gendreau, Christelle Guéret, and Andrés L. Medaglia. 2013. A review of dynamic vehicle routing problems. *European Journal of Operational Research* 225, 1 (2013), 1–11. https://doi.org/10.1016/j.ejor.2012.08.015

[14] Harilaos N. Psaraftis, Min Wen, and Christos A. Kontovas. 2016. Dynamic vehicle routing problems: Three decades and counting. *Networks* 67, 1 (2016), 3–31. https://doi.org/10.1002/net.21628

[15] Vitória Pureza and Gilbert Laporte. 2008. Waiting and buffering strategies for the dynamic pickup and delivery problem with time windows. *INFOR: Information Systems and Operational Research* 46, 3 (2008), 165–175.

[16] Stefan Ropke and David Pisinger. 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science* 40, 4 (2006), 455–472. https://doi.org/10.1287/trsc.1050.0135

[17] Alberto Santini, Michael Schneider, Thibaut Vidal, and Daniele Vigo. 2021. Decomposition strategies for vehicle routing heuristics. (2021). http://www.optimization-online.org/DB_HTML/2021/03/8316.html

[18] Carlo S Sartori and Luciana S Buriol. 2020. A study on the pickup and delivery problem with time windows: matheuristics and new instances. *Computers & Operations Research* 124 (2020), 105065.

[19] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. 2013. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers and Operations Research* 40, 1 (jan 2013), 475–489. https://doi.org/10.1016/j.cor.2012.07.018

Meta-heurística GRASP aplicada ao problema do roteamento de veículos capacitado: um estudo de caso com a empresa Loggi

Mayrla Jansen
mayrla.jansen@discente.ufma.br
Universidade Federal do Maranhão (UFMA)
Sao Luís, MA, Brasil

Glaubos Clímaco
glaubosclimaco@gmail.com
Departamento de Informática (DEINF - UFMA)
Sao Luís, MA, Brasil

ABSTRACT

In this work, a method for solving the capacitated vehicle routing problem (CVRP) is proposed. CVRP is an extension of the vehicle routing problem, which in turn derives from the classic traveling salesman problem. To solve this problem, a heuristic based on the GRASP meta-heuristic is proposed in order to obtain good quality solutions in a reasonable computational time. GRASP is an iterative algorithm composed of a construction phase for an initial solution, and another phase in which a local search takes place, promoting a refinement of this solution. To validate the proposal, computational experiments were carried out on instances generated from a real database, made available by the company Loggi.

KEYWORDS

GRASP, meta-heuristic, CVRP, Loggi

ACM Reference Format:
Mayrla Jansen and Glaubos Clímaco. 2021. Meta-heurística GRASP aplicada ao problema do roteamento de veículos capacitado: um estudo de caso com a empresa Loggi. In *PBP-LOGGI: Workshop Virtual Loggi: Resultados do Programa de Bolsas de Pesquisa, Dez 08, 2021, ?????*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUÇÃO

Com o avanço das demandas por bens e serviços, aliada ao grande crescimento da competitividade presente no mercado, se torna cada vez mais notório que os custos que antes poderiam ser relevados agora se tornam de fundamental importância para se tornarem objetos de estudo e diminuição de valores. Nesse ponto, em especial, as despesas com as operações. Essas devem ser cada vez mais pensadas do ponto de vista logístico, para que o produto final obtenha o menor valor de custo possível. Dessa forma, a utilização de Pesquisa Operacional para este fim, se torna de grande valia, visto que a aplicação de suas técnicas buscam o máximo desempenho operacional, vantagens estratégicas e contabilizam o impacto financeiro [de Araújo et al. 2018].

Nesse contexto, surge o problema de roteamento de veículos (PRV). O PRV é um problema de otimização combinatória que pergunta "Qual é o conjunto ideal de rotas para uma frota de veículos percorrer a fim de realizar entregas a um determinado conjunto de clientes?". O PRV tem muitas aplicações diretas na indústria, e o uso de *softwares* de otimização pode proporcionar economia de 5% para uma empresa [Hasle et al. 2007], já que o transporte é geralmente um componente significativo do custo de um produto (10%) [Rodrigue et al. 2016].

O PRV diz respeito ao serviço de uma empresa de entrega. As entregas são realizadas a partir de um ou mais depósitos que têm um determinado conjunto de veículos operados por um conjunto de motoristas que podem se deslocar em uma determinada rede rodoviária para um conjunto de clientes. O objetivo do PRV é determinar um conjunto de rotas, S, (uma rota para cada veículo que deve começar e terminar em seu próprio depósito) de forma que todos os requisitos dos clientes e restrições operacionais sejam atendidos e o custo global de transporte seja minimizado. Este custo pode ser monetário, distância ou outro [Toth e Vigo 2002].

A rede de estradas pode ser descrita usando um grafo onde os arcos são estradas e vértices são junções entre eles. Os arcos podem ser direcionados ou não direcionados devido à possível presença de ruas de mão única ou custos diferentes em cada direção. Cada arco possui um custo associado que geralmente é sua duração ou tempo de viagem, que pode depender do tipo de veículo [Toth e Vigo 2002]. Neste trabalho é abordado o problema de roteamento de veículos capacitado (PRVC). O PRVC é uma variante do PRV na qual os veículos possuem uma capacidade de carga limitada das mercadorias que devem ser entregues (ver Figura 1).

Figure 1: Representação em grafo de uma solução para o PRVC para 8 vértices

Determinar a solução ótima para PRVC é NP-difícil [Toth e Vigo 2002], então o tamanho dos problemas que podem ser resolvidos, de forma

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
PBP-LOGGI, Dez 08, 2021, ????
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06…$00
https://doi.org/10.1145/1122445.1122456

otimizada, usando programação matemática ou otimização combinatória pode ser limitado. Portanto, os solucionadores comerciais tendem a usar heurísticas devido ao tamanho e à frequência dos PRVC do mundo real que precisam resolver. Esse fato motivou os autores deste trabalho a proporem um método baseado na meta-heurística GRASP (*greedy randomized adaptive search prodecures)* para a resolução do problema, a partir de dados reais da empresa Loggi [LOGGI 2021].

O restante deste artigo é organizado da seguinte maneira. A Seção 2 descreve os principais trabalhos relacionados ao PRVC. Na Seção 3 é apresentado o desenvolvimento do algoritmo GRASP, assim como os algoritmos utilizados em suas fases: guloso e de busca local. A Seção 4, descreve a instanciação dos dados e a análise feita com os resultados obtidos pelo algoritmo GRASP. Finalmente, na Seção 5 é apresentada a conclusão deste trabalho.

2 TRABALHOS RELACIONADOS

O PRVC é amplamente estudado na literatura, e foi estudado pela primeira vez em um artigo de [Dantzig e Ramser 1959], no qual a primeira abordagem algorítmica foi escrita e aplicada a entregas de gasolina. Esse problema pode ser resolvido por métodos exatos (modelos matemáticos e algoritmos de branch-and-bound) ou abordagens heurísticas. Métodos exatos podem obter soluções ótimas, mas sua complexidade computacional resultará em tempo exponencial quando o tamanho do problema for grande. As abordagens heurísticas podem ser divididas em abordagens heurísticas clássicas e abordagens metaheurísticas.

Entre os procedimentos heurísticos clássicos, o algoritmo construtivo de [Clarke e Wright 1964] (CWS) é provavelmente a heurística mais citada e utilizada para resolver o PRVC. Muitas variantes e melhorias do CWS foram propostas na literatura. Por exemplo, uma generalização da definição da função de economia feita por [Mole e Jameson 1976], introduzindo dois parâmetros para controlar o comportamento da economia. Da mesma forma, um procedimento baseado no algoritmo CWS foi desenvolvido por [Holmes e Parker 1976], utilizando a mesma função de economia, mas introduzindo um esquema de perturbação de solução para evitar rotas de baixa qualidade.

Utilizando heurísticas construtivas como base, as meta-heurísticas se tornaram populares para o PRV durante os anos noventa. Esses procedimentos são capazes de encontrar soluções quase ótimas e até ótimas globais para problemas combinatórios. Alguns exemplos iniciais aplicados para resolver o PRVC são o *Tabu Route* [Gendreau et al. 1994] ou o *Boneroute* [Tarantilis e Kiranoudis 2002]. Algoritmos de busca tabu (TS), como os propostos por [Taillard 1993] ou [Toth e Vigo 2003], estão entre as meta-heurísticas mais populares. Por fim, os algoritmos genéticos (AG) também têm desempenhado um papel importante no desenvolvimento de abordagens eficazes para o PRV. Alguns exemplos são os estudos de [Berger e Barkaoui 2003] e [Mester e Bräysy 2007].

É importante ressaltar que, após a revisão bibliográfica para este trabalho, foi observado que apesar de vários trabalhos na literatura relatarem heurísticas/meta-heurísticas interessantes para a resolução do PRVC, poucos trabalhos abordam instâncias ou cenários baseados em dados geográficos reais. Em muitos casos, os autores posicionaram pontos aleatoriamente em um plano cartesiano e adotaram a distância euclidiana entre dois pontos.

3 META-HEURÍSTICA GRASP

Neste trabalho, é proposto um método de solução baseado na meta-heurística GRASP. GRASP do inglês, *Greedy Randomized Adaptive Search Procedure*, proposta por [Resende e Ribeiro 2016] e utilizada para resolver problemas complexos e de grande porte. Essa meta-heurística é executada iterativamente, e cada iteração é composta por duas fases: construção e busca local. Uma solução viável S é gerada na fase de construção, e então sua vizinhança é explorada por uma busca local, a fim de encontrar uma solução melhor. Esse processo iterativo é repetido até que um critério de parada seja alcançado, que pode ser um número limitado de iterações permitidas, um número de iterações sem melhoria da solução, entre outros. Por fim, a melhor solução encontrada em todas as iterações é retornada.

O Algoritmo 1 apresenta um pseudo-código do GRASP. Ele recebe como parâmetros o $MaxIterations$ e o α, sendo $MaxIterations$ o número de iterações a serem executadas e o α determina o quão guloso é o algoritmo, variando de zero a um, em que quanto mais próximo de zero mais aleatório e quanto mais próximo de um, mais guloso. Na linha 1, o valor da melhor solução é iniciado com um valor muito alto (∞), em seguida, das linhas 2 a 8, são executadas $MaxIterations$ iterações, no qual uma solução é construída de maneira gulosa e refinada com uma busca local. A cada iteração é feito um teste que verifica se a solução obtida é melhor que a melhor solução encontrada até o momento, e caso positivo, o valor da melhor solução é atualizado.

Algorithm 1 GRASP ($\alpha, MaxIterations$)

1: $custo(melhorSolucao) \leftarrow \infty$
2: **for** $i = 0, 1, \ldots, MaxIterations$ **do**
3: $\quad solucaoGulosa \leftarrow BuscaGulosa(\alpha)$
4: $\quad solucao \leftarrow BuscaLocal(solucaoGulosa)$
5: $\quad$ **if** $custo(solucao) < custo(melhorSolucao)$ **then**
6: $\quad\quad melhorSolucao \leftarrow solucao$
7: $\quad$ **end if**
8: **end for**
9: **return** $bestSolution$

3.1 Algoritmo de busca gulosa

Para a fase de construção do GRASP, uma busca gulosa foi implementada considerando as seguintes condições do problema: i) um vértice (cliente) deve ser visitado por apenas um único veículo que deve ter capacidade maior ou igual à quantidade de entregas (demanda) desse vértice; ii) todos os veículos da frota partem de um mesmo vértice origem (depósito); e iii) cada vértice deve ser visitado exatamente uma vez.

Diferentemente de um algoritmo guloso tradicional, geralmente aplicado em situações de tomada de decisão, neste trabalho foi implementado uma busca gulosa mais complexa. Tal busca não considera simplesmente a menor distância de um vértice para outro, mas sim uma lista de candidatos restrita (LCR); e para pertencer a essa LCR, o custo incremental da visita de um vértice por determinado veículo deve considerar um valor limite, estipulado pela Equação (1).

$$limite = menor_distancia + \alpha * (maior_distancia - menor_distancia) \quad (1)$$

Nesse caso, a LCR será composta por vértices que ainda não foram visitados e que possuam um custo incremental menor que o valor limite da Equação (1). Mais detalhes sobre a busca gulosa implementada são apresentados no pseudo-código do Algoritmo 2.

Algorithm 2 BuscaGulosa $(\alpha, n_veiculos, n_vertices)$

```
S ← ∅                                          ▷ inicializando a solução
for k = 0 até k < n_veiculos do
    LC ← i ∈ V : i é um vértice ainda não visitado
    LRC ← criar_LRC(α, LC, k)                  ▷ a partir da Equação (1)
    v ← random(LRC) ▷ escolhendo um vértice para ser visitado por k
    S[k] ← S[k] ∪ v
end forreturn S
```

No Algoritmo 2, iteramos tanto sobre os veículos quanto sobre os vértices, de modo a obter soluções considerando a quantidade de veículos e suas capacidades. Para cada veículo, inicialmente são considerados todos os vértices ainda não visitados (conjunto LC). Em seguida, é cria uma lista de candidatos restrita (LCR), que é composta pelos vértices que obtêm custo incremental menor ou igual ao *limite* e que a demanda não ultrapassa a capacidade atual do veículo. Após a lista finalizada, aleatoriamente é escolhido um dos vértices de LRC que será incorporado à solução gulosa que está sendo montada. Ao final do laço que percorre todos os veículos, a solução construída é retornada.

3.2 Algoritmo de busca local

Buscas locais, também referidas na literatura como buscas na vizinhança, são procedimentos utilizados para melhorar uma solução viável obtida, por exemplo, através de uma heurística construtiva. Esses métodos são mais genéricos e podem auxiliar uma ampla gama de problemas sem a necessidade de uma compreensão mais profunda do mesmo [Sauer 2007].

Após a fase de construção, já é possível ter uma solução gulosa, contudo, como foi visto anteriormente, essa solução apresenta desvantagens por não considerar o problema como um todo. Com a busca local, é realizado um refinamento na solução obtida, de maneira a melhorar a qualidade da solução construída. Ou seja, com essa fase, pretende-se reduzir o custo das rotas, caso seja possível. O Algoritmo 3 é um pseudo-código da busca local realizada neste estudo de caso.

Para este trabalho a busca local ocorreu da seguinte forma. Para cada veículo e para cada vértice pertencente à rota desse veículo, é verificado se uma troca de ordem de visita entre dois vértices melhora o custo da solução. Caso melhore, a solução refinada da busca local é atualizada. Por fim, a solução $S_refinada$ é retornada.

4 EXPERIMENTOS COMPUTACIONAIS

Para a realização dos experimentos deste trabalho, a linguagem escolhida foi o python3.8 e o ambiente foi um computador pessoal Intel® Core™ i5 CPU@1.60GHz com 8GB RAM, utilizando o sistema operacional Linux Ubuntu 18.04.5.

Algorithm 3 BuscaLocal (S)

```
S_refinada ← S
for k = 0 até k < n_veiculos do
    for v = 0 até v < tamanho(S[k]) − 1 do
        custo_k_antes ← custo(S[k])
        troca(S[k][v], S[k][v + 1])  ▷ trocando a ordem de visita na
rota k
        custo_k_depois = custo(S[k])
        if custo_k_depois ≥ custo_k_antes then
            troca(S[k][v + 1], S[k][v])          ▷ desfazendo a troca
        else
            S_refinada ← S
        end if
    end for
end for
return S_refinada
```

4.1 Criação das instâncias

Os dados referentes aos pontos de entrega (longitude e latitude), quantidade de entrega por pedido e capacidade dos veículos, cedidos pela empresa Loggi para análise e testes, estavam em formato *JSON* [LOGGI 2021], e para fins desse estudo a quantidade de pontos usada foi de 100 e 200. *JSON* é um formato de texto derivado da linguagem Javascript sendo representado por uma coleção de objetos compostos por pares de chave-valor [Silva e Silva Júnior 2018].

Para fins de implementação, optou-se por transformar os dados presentes no JSON, em um novo arquivo no formato `.txt`, tal que, nesse novo arquivo, estaria elencado a quantidade de pontos de entregas, capacidade dos veículos, quantidade de pedidos e por fim, uma matriz que representa a distância de um ponto para os demais como mostrado na Figura (2). Esse arquivo será o grafo, usado na entrada para o algoritmo GRASP, pois nele conterá todas as informações da instância.

A capacidade dos veículos foi dada com base no total das demandas. Considerando K o número de veículos e Q a capacidade dos veículos, a razão (soma de todas as demandas)/$K \times Q$ deve ser o mais próximo possível de 1, tendo como consequência o número mínimo de veículos para se obter uma solução viável.

Figure 2: Exemplo de instância com cinco pontos de entrega

Para construir a matriz de distância entre os pontos, foi empregada a Distance Matrix API do Google [GOOGLE 2021]. O Algoritmo (4) mostra o pseudo-código da utilização dessa API neste

estudo de caso. A API é conectada com o projeto (linha 1) e necessita dos seguintes parâmetros: longitude e latitude do ponto X e do ponto Y, onde X é o ponto de origem e Y é o ponto de destino (linha 2); e a partir dessas coordenadas, a API retorna um conjunto de dados sobre esses pontos. Para este trabalho foi necessário apenas registrar a distância real entre esses pontos (linhas 3 e 4).

Algorithm 4 GetDistance

1: $gmaps \leftarrow googlemaps.Client(key =' xxx')$
2: $getDistance \leftarrow gmaps.distance_matriz(X, Y)$
3: $distance \leftarrow getDistance['distance']['text']$
4: $distance \leftarrow (distance.replace(",",".") * 1000)$
5: **return** $distance$

Vale ressaltar que o retorno da API foi padronizado para a unidade de medida em quilômetros, de modo a facilitar a manipulação dos dados, visto que assim, todos estão na mesma unidade de medida. E a distância de um ponto para ele mesmo, foi padronizada em 10000.0, com o intuito de representar uma distância excessiva para que sempre fosse excluída nos algoritmos de construção de rotas, mostrando que não se deve ir de um ponto para ele mesmo.

4.2 Análise dos resultados

Para os fins deste trabalho, foi avaliado tanto se as soluções atendiam os critérios, quanto se o tempo de solução era viável. Além disso, comparou-se os resultados obtidos do algoritmo GRASP com os resultados obtidos utilizando apenas o algoritmo de Busca Gulosa. Nessa comparação, foi considerado que, geralmente, quando não se faz uso de metaheurísticas, um algoritmo guloso é o mais utilizado por empresas ao tomar decisões acerca do PRVC.

Iniciou-se realizando testes com o GRASP para refinar os parâmetros utilizados e obter a melhor proporção de menor custo x tempo computacional aceitável. Tal que, para este trabalho o custo representa a distância total a ser percorrida para que todos os vértices sejam visitados. Logo após, utilizou-se a mesma matriz de distância para computar os resultados obtidos com a Busca Gulosa.

Os resultados para esta fase inicial de refinamento de parâmetros, neste caso, o parâmetro α, estão dispostos na Tabela 1. Sendo a quarta coluna referente à quantidade de iterações dentro do GRASP fixa em 10 e o α utilizado para a fase de construção que varia entre 0.70, 0.80 e 0.90. Da quinta a sétima coluna é mostrado os resultados e tempo referente ao algoritmo GRASP e a oitava e nona mostram os resultados obtidos a partir da Busca Gulosa. Outro ponto importante é frisar que o tempo médio está em milissegundos.

Com base nos primeiros resultados obtidos, na maioria dos casos, o $\alpha = 0.7$ mostra resultados bem melhores em relação aos outros parâmetros. Além disso, é perceptível que os resultados obtidos com o GRASP são bem mais otimizados que os resultados da Busca Gulosa, em um tempo computacional similar. Por exemplo, na instância PA com 200 pontos, o GRASP empregou uma redução de aproximadamente 16% no custo da solução.

Contudo, ainda existem formas de melhorar o resultado obtido com o algoritmo GRASP, por exemplo, refinando o parâmetro de iterações ($MaxIterations$). Os resultados para o refinamento das iterações estão dispostos na Tabela 2.

				GRASP			Busca Gulosa	
Qtd. Pontos	Q	**UF**	**Par.**	**Custo médio**	**Menor custo**	**Tempo médio**	**Menor custo**	**Tempo médio**
100	50	RJ	It. = 10, α = 0,70	190,2	186,5	0,54	229,7	0,06
		DF		1.123,6	1.118,5	0,50	1.444,2	0,07
		PA		536,1	534,3	0,59	611,1	0,06
200	100	RJ		720,9	719,2	1,82	854,1	0,20
		DF		1.054,0	1.050,9	1,82	1.751,4	0,20
		PA		1.252,6	1.250,0	1,93	1.503,4	0,21
100	50	RJ	It. = 10, α = 0,80	200,3	199,5	0,53	226,8	0,07
		DF		992,5	990,7	0,49	1.584,2	0,06
		PA		556,0	553,2	0,53	1.464,2	0,08
200	100	RJ		762,1	759,9	1,80	973,0	0,20
		DF		1.309,7	1.308.2	1,75	1.677,1	0,19
		PA		1.239,4	1.238,7	1,89	1.527,0	0,22
100	50	RJ	It. = 10, α = 0,90	212,3	210,6	0,66	269,9	0,08
		DF		1.193,0	1.191,4	0,51	1.368,2	0,06
		PA		575,9	574,3	0,64	694,3	0,05
200	100	RJ		858,7	857,1	1,87	999,2	0,21
		DF		1.394,3	1.392,1	1,76	1.498,0	0,24
		PA		1.366,2	1.364,2	1,93	1.554,7	0,23

Table 1: Resultados de calibragem do parâmetro α (GRASP) X resultados da Busca Gulosa

		GRASP (Melhor Caso)		
Qtd. pontos	**UF**	**Iteração = 10**	**Iteração = 20**	**Iteração = 30**
100	RJ	186,5	186,5	184,3
	DF	1.118,5	922,7	878,6
	PA	534,3	537,8	534,0
200	RJ	719,2	723,6	699,9
	DF	1.050,9	1.077,7	1.069,1
	PA	1.250,0	1.216,9	1.150,9

Table 2: Tabela de resultados dos testes do refinamento das iterações (GRASP

Com esses novos resultados ficam claros que o aumento da quantidade de iterações, torna os resultados cada vez melhores. No entanto, a relação de aumento de interações é diretamente proporcional ao aumento do tempo computacional. Dessa maneira, a solução de melhor qualidade que possui um custo computacional aceitável, fica com os parâmetros: α igual a 0.70 e quantidade de iterações igual a 30.

5 CONCLUSÃO

Neste trabalho aplicamos a meta-heurística GRASP para a resolução do Problema de Roteamento de Veículos Capacitado. Para a validação da proposta de solução, foram geradas instâncias a partir de uma base de dados real fornecida pela empresa Loggi, contendo as informações: quantidade de pontos, demandas, e matriz de distâncias.

O método proposto neste trabalho mostrou-se como viável para resolução do problema do roteamento de veículos capacitado. Apresentou-se resultados satisfatórios e conseguiu-se atender todos os

critérios elencados tanto pelo problema, quanto pela própria técnica e os recursos computacionais.

A grande vantagem do método proposto está em sua facilidade de adaptação a diversos projetos que busquem esta resposta ao PRVC. Entretanto, a API do Google utilizada para cálculo das distâncias não é gratuita, o que gerou algumas dificuldades na geração das instâncias.

Ao se resolver o problema abordado, "manualmente" ou por algum algoritmo mais simples que o GRASP, como um algoritmo puramente guloso, pode-se correr o risco de facilmente cair em ótimos locais. Nos experimentos computacionais realizados, foi observado que o GRASP é bem mais robusto nesse sentido. Desse modo, acredita-se que a solução apresentada neste trabalho pode auxiliar o setor logístico de empresas, a fim de se ter um melhor controle sobre a operação de transporte com base em dados reais.

É fundamental a realização de estudos complementares sobre a técnica e o problema. Além disso, é de suma importância que essas técnicas sejam cada vez mais incorporadas no âmbito empresarial, de modo a testá-las no contexto de uso diário para que assim, a técnica possa ser cada vez mais aperfeiçoada.

Como trabalhos futuros, dado que o GRASP é um método multi-partida e "sem memória", pois informações de uma iteração não são transmitidas a iterações futuras, pode-se combinar o GRASP proposto com mineração de dados para que se possa re-utilizar informações obtidas em iterações passadas [Santana et al. 2020]. Neste trabalho foi explorado apenas uma vizinhança na fase de busca local, então um caminho natural para investigações futuras seria elaborar um maior número de vizinhanças e combiná-las em um algoritmo VND (*Variable Neighborhood Descent*) [Clímaco et al. 2021].

REFERENCES

[Berger e Barkaoui 2003] Berger, J. e Barkaoui, M. (2003). A hybrid genetic algorithm for the capacitated vehicle routing problem. In *Genetic and evolutionary computation conference*, pages 646–656. Springer.

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

[Clímaco et al. 2021] Clímaco, G., Rosseti, I., Silva, R., e Guerine, M. (2021). Reactive grasp for the prize-collecting covering tour problem. *RAIRO: Recherche Opérationnelle*, 55:1441.

[Dantzig e Ramser 1959] Dantzig, G. B. e Ramser, J. H. (1959). The truck dispatching problem. *Management science*, 6(1):80–91.

[de Araújo et al. 2018] de Araújo, F., Lima, A. A., e Lima, M. d. A. C. (2018). Otimização de rota e redução dos custos logísticos: estudo de caso em uma empresa de contabilidade. *Brazilian Journal of Development*, 4(1):136–144.

[Gendreau et al. 1994] Gendreau, M., Hertz, A., e Laporte, G. (1994). A tabu search heuristic for the vehicle routing problem. *Management science*, 40(10):1276–1290.

[GOOGLE 2021] GOOGLE (2021). Distance matrix api. https://console.cloud.google.com/marketplace/product/google/distance-matrix-backend.googleapis.com?q=search&referrer=search. Acesso 07 de outubro de 2021.

[Hasle et al. 2007] Hasle, G., Lie, K.-A., e Quak, E. (2007). *Geometric modelling, numerical simulation, and optimization.* Springer.

[Holmes e Parker 1976] Holmes, R. e Parker, R. (1976). A vehicle scheduling procedure based upon savings and a solution perturbation scheme. *Journal of the Operational Research Society*, 27(1):83–92.

[LOGGI 2021] LOGGI (2021). Github loggi. https://github.com/loggi/loggibud. Acesso 08 de outubro de 2021.

[Mester e Bräysy 2007] Mester, D. e Bräysy, O. (2007). Active-guided evolution strategies for large-scale capacitated vehicle routing problems. *Computers & operations research*, 34(10):2964–2975.

[Mole e Jameson 1976] Mole, R. e Jameson, S. (1976). A sequential route-building algorithm employing a generalised savings criterion. *Journal of the Operational Research Society*, 27(2):503–511.

[Resende e Ribeiro 2016] Resende, M. G. e Ribeiro, C. C. (2016). *Optimization by GRASP.* Springer.

[Rodrigue et al. 2016] Rodrigue, J.-P., Comtois, C., e Slack, B. (2016). *The geography of transport systems.* Routledge.

[Santana et al. 2020] Santana, Í., Plastino, A., e Rosseti, I. (2020). Improving a state-of-the-art heuristic for the minimum latency problem with data mining. *International Transactions in Operational Research.*

[Sauer 2007] Sauer, J. G. (2007). *Abordagem de Evolução diferencial híbrida com busca local aplicada ao problema do caixeiro viajante.* PhD thesis, Master's thesis, Pontificia Universidade Católica do Paraná.

[Silva e Silva Júnior 2018] Silva, P. C. d. e Silva Júnior, J. B. d. (2018). Análise da representação semântica de modelos de dados do formato json. *Revista de Sistemas e Computação-RSC*, 8(1).

[Taillard 1993] Taillard, É. (1993). Parallel iterative search methods for vehicle routing problems. *Networks*, 23(8):661–673.

[Tarantilis e Kiranoudis 2002] Tarantilis, C. D. e Kiranoudis, C. T. (2002). Using a spatial decision support system for solving the vehicle routing problem. *Information & Management*, 39(5):359–375.

[Toth e Vigo 2002] Toth, P. e Vigo, D. (2002). The vehicle routing problem monographs on discrete mathematics and applications siam. *Philadelpia, Pennsylvania.*

[Toth e Vigo 2003] Toth, P. e Vigo, D. (2003). The granular tabu search and its application to the vehicle-routing problem. *Informs Journal on computing*, 15(4):333–346.

Estudo de Caso da Meta-heurística GRASP Aplicada ao Problema de Alocação de Hubs Capacitados

Nathasha Pinto
nathasha.pinto@discente.ufma.br
Universidade Federal do Maranhão (UFMA)
São Luís, MA, Brasil

Glaubos Clímaco
francisco.glaubos@ufma.br
Departamento de Informática (DEINF - UFMA)
São Luís, MA, Brasil

ABSTRACT

Em setores que exigem uma logística exemplar, como por exemplo em uma empresa de entregas, problemas de otimização são comuns. O fluxo de entrada e saída de objetos é muito grande e por isso é necessário se utilizar de métodos heurísticos que minimizam os custos de transporte. O presente artigo descreve um estudo de caso utilizando a meta-heurística GRASP no problema de localização de hubs capacitados, comparando seu desempenho ao de um algoritmo puramente guloso, buscando assim minimizar o custo total de deslocamento e verificar qual a melhor abordagem de solução. O problema foi adaptado pensando em seu uso real, no qual as instâncias utilizadas para os testes computacionais foram geradas a partir de uma base de dados real.

KEYWORDS

GRASP, meta-heurística, PpHC, Loggi

Copyright:
Nathasha Pinto and Glaubos Clímaco. 2021. Estudo de Caso da Meta-heurística GRASP Aplicada ao Problema de Alocação de Hubs Capacitados. In *PBP-LOGGI: Workshop Virtual Loggi: Resultados do Programa de Bolsas de Pesquisa, Dez 08, 2021,* . LOGGI, São Paulo, SP, BR, 4 pages.

1 INTRODUÇÃO

Problemas de otimização consistem em achar a melhor combinação dentre um conjunto de variáveis para maximizar ou minimizar uma função, geralmente chamada de função objetivo ou função custo [3]. Esses problemas de otimização no contexto de uma rede de entregas são frequentes, uma vez que o fluxo de entrada e saída é muito grande. Há diversos métodos exatos que em teoria podem ser aplicados para a resolução de tais problemas, mas na prática, ao serem expostos a instâncias de grande porte se tornam complexos e inviáveis. Para isso foram criadas meta-heurísticas, que são estruturas algorítmicas independentes que fornecem um conjunto de diretrizes ou estratégias para desenvolver algoritmos de otimização e que podem gerar soluções viáveis para problemas reais [9].

Quando a transmissão de dados entre os pontos não se dá de forma direta, essa rede é chamada de *hub-and-spoke* [2], e os problemas existentes nessa área se dão justamente na comunicação entre os pontos não-hubs e os hubs, pois é necessário avaliar qual o melhor meio para que essa transmissão ocorra, considerando o custo mínimo. A partir disso surge o problema de alocação de hubs (PpH), problema que está inserido na classe NP-difícil [8], cujo o objetivo é encontrar a melhor localização dos hubs e alocar pontos não-hubs à eles, minimizando uma função objetivo que descreve o fluxo intercambiado e seu custo.

Sendo assim, neste trabalho será abordado um estudo de caso sobre o problema de alocação de p-hubs capacitado (PpHC). Esse problema consiste em uma variante do problema formulado por Morton E O'kelly [12] e tem como parâmetros que diferem do principal problema: um número fixo de p hubs, a capacidade máxima dos hubs selecionados, a restrição dos pontos (o ponto só pode se ligar a um hub), e o custo de configuração associado.

No intuito de resolver um problema real, o presente estudo utiliza instâncias reais advindas de uma base de dados fornecida pela LOGGI, uma empresa de entregas. E nessas instâncias será aplicada uma heurística baseada em GRASP (Greedy Randomized Adaptive Search Procedures) [7]. Com o GRASP será obtida uma solução em duas fases; a primeira será a aplicação de um algoritmo guloso para a construção de uma solução, e em seguida será feita uma busca local nessa solução, a fim de aprimorá-la.

2 O PPHC NA LITERATURA

O problema de alocação de hubs não é um problema recente tendo sua primeira solução proposta por Morton E O'kelly [13]. O modelo resultante tem uma função objetivo quadrática não convexa e restrições lineares. A primeira formulação linear foi desenvolvida por James F Campbell [4], permitindo o uso de software de programação matemática padrão.

Em trabalhos mais recentes como o de Lüer-Villagra et al. [11], é sugerido o uso de uma combinação de algoritmo genético com outro modelo matemático aplicado, fazendo uso de até 200 pontos em seus testes. A conclusão obtida é que a meta-heurística aplicada permite a resolução de pequenas e médias instâncias de forma rápida e satisfatória, mas quando aplicada a grandes instâncias, se torna custosa.

Para a resolução do PpH, a maioria dos pesquisas fazem uso de heurísticas híbridas, como mostra [15] que faz uso de VNS juntamente com TS (Tabu Search) que podem ser aplicados em cenários reais. Já os autores Alkaabneh et al. [1] fazem uma comparação entre o uso de GRASP e de uma relaxação Lagrangiana, mostrando que ambas produzem resultados de alta qualidade, com destaque para o GRASP que foi capaz de gerar soluções ótimas em quase 90% dos testes.

Um estudo realizado por de Carvalho [6] apresenta uma proposta de solução utilizando uma hibridização de meta-heurísticas para o problema que será analisado no atual trabalho, o PpHC. O autor traz um método nomeado ILS-RVND-PR que é um fusão das técnicas

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
PBP-LOGGI, Dez 08, 2021,
© 2021 Workshop Virtual Loggi

Iterated Local Search [10], Random Variable Neighborhood Descent [16] e Path-Relinking [14], que mostra que a aplicação foi capaz de encontrar soluções rápidas em pouco tempo.

Como foi observado, muitos autores propõem abordagens distintas buscando encontrar a solução mais razoável para o problema. Deste modo os autores do presente trabalho propõem o uso da meta-heurística GRASP, para resolver o PpHC a partir de instâncias inéditas baseadas em dados reais, a fim de verificar a sua viabilidade de aplicação na indústria.

3 MÉTODO DE SOLUÇÃO

Neste trabalho, propomos uma heurística baseada na meta-heurística GRASP proposta por Feo e Resende [7], que é um método iterativo constituído por duas fases: a de construção e a de busca local.

Na fase de construção, elementos que podem compor uma solução são inseridos em uma lista de candidatos (LC), que então é ordenada de acordo com a contribuição do elemento na função objetivo. Em seguida, os melhores elementos de LC são escolhidos para compor uma lista de candidatos restrita (LCR). Por fim, escolhe-se, aleatoriamente, um elemento de LRC para compor a solução. Esse processo é repetido até que uma solução viável para o problema seja obtida.

De acordo com Feo e Resende [7], não há garantias que a solução construída seja localmente ótima. Portanto, a partir da solução construída, é realizado um procedimento de refinamento conhecido como Busca Local.

O Algoritmo 1 apresenta um pseudo-código do método GRASP, no qual *Seed* representa uma semente para as componentes aleatórias dentro do método, e *MaxIterations* é a quantidade de iterações que o GRASP irá executar. As listas LC e LCR serão geradas com a função *GreedyRandomizedConstruction*, e delas será gerada uma solução parcial. Já a busca local é feita na função *LocalSolution*, para refinar a solução.

Algorithm 1 Pseudoalgoritmo GRASP

```
function GRASP(MaxIterations, Seed)
    for k = 0, ..., MaxIterations do
        Solution ← GreedyRandomizedConstruction(Seed)
        Solution ← LocalSolution(Solution)
        UpdateSolution(Solution, BestSolution)
    end for
     return BestSolution
end function
```

3.1 Fase de Construção

A fase de construção desenvolvida neste trabalho, difere da construção tradicionalmente feita com o GRASP. Na nossa proposta, inicialmente são selecionados p pontos da rede para serem hubs, e em seguida, são realizadas ligações de pontos não-hubs (V) a hubs (H), de forma que minimize o custo da solução, que se define pela soma total do custo de todas as ligações entre hubs e não-hubs. Por fim, essa fase retorna uma solução $S(H, V)$ contendo associações entre hubs e não-hubs. O Algoritmo 2 mostra com mais detalhes como são realizadas essas associações

Os parâmetros de decisão para escolha do hub são: proximidade e capacidade. Uma vez verificado qual o mais próximo, precisa-se conferir se a demanda do vértice é suportada pela capacidade do hub.

Algorithm 2 Algoritmo que liga os vértices aos hubs

```
A partir de V escolher aleatoriamente p hubs para compor H
V̄ ← V
while V̄ ≠ ∅ do
    escolher o v ∈ V̄ mais próximo a um hub h ∈ H
    v.hub = h   ▹ Associando v ao hub h, caso a capacidade de h
seja respeitada
    Atualizar a capacidade do hub h
    V̄ ← V̄ \ v
end while
return S
```

3.2 Fase da busca local

O pseudo-código de busca local aplicado a partir de uma solução construída S é representado pelo Algoritmo 3. Para cada vértice não-hub é feita uma tentativa de troca de associação com um diferente hub, e caso essa troca resulte em uma solução de menor custo, essa nova associação é estabelecida, e as capacidades dos hubs envolvidos na troca são atualizadas.

Algorithm 3 Busca_Local(S)

```
Hubs[]
melhor_custo = custo(S)
melhor_solucao = S
S′ = S
for cada v ∈ V′ do
    for cada h ∈ H′ do
        if h = v.hub then
            Continue ▹ Nada é feito, e o próximo v é considerado
        end if
        v.hub = h          ▹ associando vértice v a um outro hub
        Atualizar a solução S′   ▹ atualiza a solução caso a troca
seja viável
        if custo(S') < custo(S) then
            melhor_custo = custo(S′)
            melhor_solucao = S′
        else
            desfazer a associação
        end if
    end for
end for
return melhor_solucao
```

4 EXPERIMENTOS COMPUTACIONAIS

Nesta sessão serão apresentados os resultados obtidos com a aplicação do método proposto, e os seus resultados serão comparados com resultados obtidos a partir do algoritmo guloso 2. Neste trabalho, supomos que em situações reais de tomada de decisão acerca do PpHC, geralmente utiliza-se estratégias gulosas para a alocação de hubs.

Todos os códigos-fonte foram implementados na linguagem de programação Python3, em um computador pessoal Intel® Core™ i3 CPU@2.30GHz com 8GB RAM, utilizando o sistema operacional Linux.

4.1 Instâncias do estudo de caso

Para a realização dos experimentos computacionais, considerou-se um estudo de caso que se constitui de instâncias geradas a partir de uma base de dados real da empresa Loggi [?]. A base de dados utilizada contém localizações reais distribuídas em três estados: Pará, Distrito Federal e Rio de Janeiro. Foram extraídos dessa base o número de hubs que cada estado contém, as instâncias que indicavam a coordenada geográfica, e a demanda de cada ponto (ou cliente). As instâncias geradas variam de tamanho, tendo a menor 50 pontos e a maior 200, parâmetro definido com bases nos trabalhos anteriormente analisados [6, 11, 15].

Após essa extração foi necessário descobrir as distâncias entre todos os pontos, uma vez que o método de solução proposto necessita dessa informação. Para descobrir a localização real dos pontos e a distância entre eles, utilizou-se a Distance Matrix API fornecida pela Google. Uma vez que ela faz essa verificação, ela retorna o valor da distância em metros, incluindo do ponto para ele mesmo. Em seguida, as distâncias foram convertidas para valores em quilômetros, e visto que todos os valores são comparados inclusive do ponto para ele mesmo, foi necessário modificar esse valor para um valor alto (10000), eliminando assim a possibilidade de que o algoritmo considere um ponto mesmo como opção de ligação para si mesmo.

Essas distâncias juntamente com a demanda de cada nó, foram escritas em um arquivo `.txt`, organizados da forma como mostra o exemplo na (Figura 1), contendo a quantidade de vértices, a demanda de cada um e por fim a matriz distância criada pela API.

```
5 //Quantidade de nós//
9
2
8
8
1
1000.0          18.4            18.6            17.3            22.7
18.4            1000.0          0.2             7.7             40.7
18.6            0.2             1000.0          7.9             40.9
17.3            7.7             7.9             1000.0          33.5
23.3            41.2            41.4            34.1            1000.0
```

Figure 1: Arquivo TXT ultilizado

Os hubs considerados no problema possuem uma capacidade de atendimento de demandas, porém tal capacidade não é fornecida pela base de dados da LOGGI, então usou-se uma fórmula matemática para atender com menor custo (sem desperdício de espaço), o tamanho de cada hub. A Fórmula 1 se dá pela soma da capacidade de todos os pontos (c) dividida pela multiplicação da quantidade de hubs (h) e de um valor x que representa a capacidade que será definida ao hub. O resultado da operação é a taxa de dificuldade (tx) que o algoritmo pode obter. Essa taxa de dificuldade é dada com valores entre 0 e 1, e quão mais próximo de 1 o valor obtido pela fórmula for, mais difícil a resolução do problema será. Para o calculo de todas as capacidades das instâncias utilizadas, assumiu-se $tx = 0,9$.

$$\frac{c}{h * x} = tx \tag{1}$$

As capacidades dos hubs foram adicionadas manualmente ao algoritmo, alterando-se de acordo com a variação dos tamanhos das instâncias: 50, 100, 150, 200. A Figura 2 ilustra uma solução para uma instância do PpHC, na qual são mostrados os hubs e os pontos ligados a eles juntamente com o custo total.

Figure 2: Solução gerada a partir da instância de 50 pontos do estado do Pará

4.2 Resultados numéricos

A Tabela 1 mostra os resultados obtidos dos testes realizados com GRASP e um algoritmo guloso, exceto informações de quais hubs foram escolhidos na execução e quais pontos estão ligados a ele. Na primeira coluna é exibido o tamanho da instância (quantidade de pontos), e nas colunas seguintes, a quantidade de hubs, a média de custos, a média de tempo e o melhor custo dentre dez execuções do algoritmo GRASP. Nas colunas seis e sete, são exibidos o custo e o tempo utilizando pelo algoritmo guloso. Cada estado verificado contém quatro tamanhos diferentes de instâncias que variam entre 50 e 200. O tempo de execução é dado em segundos, e o custo em quilômetros, que se trata da soma de todas as distâncias percorridas entre cada hub e seus pontos associados.

Como verificado na Tabela 1, a quantidade de pontos (clientes) para atender influencia diretamente no custo total e na média de tempo gasto. Quanto mais cliente para atender, mais ligações são feitas entre pontos não-hubs e hubs.

Considerando os três estados, temos que o PA e DF apresentam os maiores custos, enquanto o RJ apresenta os menores custos. Como nas instâncias as distâncias entre dois pontos não costumam variar muito, atribuímos essa diferença de valor de solução à densidade da rede. As instâncias de RJ são mais densas que as demais, com mais rodovias e opções de ligação entre pontos não-hubs e hubs, o que pode causar também um aumento no tempo de processamento.

Analisando os resultados da meta-heurística e do algoritmo guloso pode-se notar que há grande diferença nos resultados. Em relação ao custo da solução obtida, o método GRASP foi superior em todos os casos, utilizando um tempo computacional similar ao do algoritmo guloso. Por exemplo, nos valores referentes ao Distrito Federal, ao compará-los na menor instância (50 pontos) há um decréscimo de mais de 90% no custo total ao utilizar o GRASP.

Tam. Instância	Q. Hubs	Média Custos	Média Tempo	Melhor Custo	Tempo	Custo
PA		GRASP			ALGORITMO GULOSO	
50	2	183,18	0,02647	177,1	0,00587	183,0
100		382,31	0,12620	365,2	0,01257	381,4
150		914,78	0,27501	886,6	0,04531	898,5
200		1802,546	0,45853	1752,9	0,08646	1820,2
DF		GRASP			ALGORITMO GULOSO	
50	3	106,18	0,04905	40,5	0,00811	467,2
100		302,53	0,11353	119,8	0,01103	796,3
150		570,25	0,24586	360,5	0,04421	1149,2
200		857,81	0,42185	748,8	0,05904	1338,7
RJ		GRASP			ALGORITMO GULOSO	
50	7	5,24	0,01755	4,0	0,01093	48,9
100		10,35	0,06192	6,1	0,09646	123,7
150		25,65	0,14467	14,2	0,03378	260,1
200		42,86	0,23688	31,4	0,04868	506,9

Table 1: Tabela de resultados dos testes com GRASP e Algoritmo Guloso

5 CONCLUSÃO

Problemas de otimização são cada vez mais comuns, ainda mais inseridos em uma rede de entregas, como a empresa Loggi. A entrada e saída de mercadorias tem um fluxo muito grande, havendo sempre que melhorar a logística do negócio. Sabe-se que buscar sempre os menores custos é uma forma de obter lucro e crescer a companhia, por conta disso nesse trabalho foi proposto o uso da meta-heurística GRASP adaptada ao problema de localização de hubs capacitados (PpHC), que visa minimizar o custo total do fluxo de mercadorias.

Utilizou-se uma base de dados da empresa com pontos reais, conferindo assim uma aplicação real do trabalho. Os resultados foram gerados a partir de testes em doze instâncias de tamanhos variados e apresentados em uma tabela mostrando o tempo médio, o custo médio, e o melhor custo de cada execução separados por estado. O trabalho mostrou bons resultados para o problema, conseguindo atingir soluções satisfatórias quando comparadas a um algoritmo puramente guloso, em um tempo computacional interessante.

Como trabalhos futuros pretende-se buscar na literatura formas de melhorar o algoritmo tanto em relação ao tempo quanto ao custo, e para isso, técnicas como Mineração de Dados e Reconexão de Caminhos devem ser analisadas [5]. Além disso, pretende-se também desenvolver uma interface gráfica que facilite a utilização do método por meio de mapas interativos.

REFERENCES

[1] Faisal Alkaabneh, Ali Diabat, and Samir Elhedhli. 2019. A Lagrangian heuristic and GRASP for the hub-and-spoke network system with economies-of-scale and congestion. *Transportation Research Part C: Emerging Technologies* 102 (2019), 249–273.

[2] Turgut Aykin. 1994. Lagrangian relaxation based approaches to capacitated hub-and-spoke network design problem. *European Journal of Operational Research* 79, 3 (1994), 501–523.

[3] José Carlos Becceneri. 2008. Meta-heurísticas e Otimização Combinatória: Aplicações em Problemas Ambientais. *INPE, Sao José dos Campos* (2008).

[4] James F Campbell. 1992. Location and allocation for distribution systems with transshipments and transportion economies of scale. *Annals of operations research* 40, 1 (1992), 77–99.

[5] Glaubos Clímaco, Isabel Rosseti, Luidi Simonetti, and Marcos Guerine. 2019. Combining integer linear programming with a state-of-the-art heuristic for the 2-path network design problem. *International Transactions in Operational Research* 26, 2 (2019), 615–641.

[6] Rodrigo de Carvalho. 2017. Heurísticas paralelas aplicadas a problemas de alocação de concentradores. (2017).

[7] Thomas A Feo and Mauricio GC Resende. 1995. Greedy randomized adaptive search procedures. *Journal of global optimization* 6, 2 (1995), 109–133.

[8] Michael R Garey and David S Johnson. 1979. *Computers and intractability*. Vol. 174. freeman San Francisco.

[9] Fred W Glover and Gary A Kochenberger. 2006. *Handbook of metaheuristics*. Vol. 57. Springer Science & Business Media.

[10] Helena R Lourenço, Olivier C Martin, and Thomas Stützle. 2010. Iterated Local Search: Framework and Applications. In *Handbook of Metaheuristics*. Springer, 363–397.

[11] Armin Lüer-Villagra, Horst A Eiselt, and Vladimir Marianov. 2019. A single allocation p-hub median problem with general piecewise-linear costs in arcs. *Computers & Industrial Engineering* 128 (2019), 477–491.

[12] Morton E O'kelly. 1986. The location of interacting hub facilities. *Transportation science* 20, 2 (1986), 92–106.

[13] Morton E O'kelly. 1987. A quadratic integer program for the location of interacting hub facilities. *European journal of operational research* 32, 3 (1987), 393–404.

[14] Colin Reeves. 1997. Modern heuristics techniques for combinatorial problems. *Nikkan Kogyo Shimbun* (1997).

[15] Ornurai Sangsawang and Sunarin Chanta. 2020. Capacitated single-allocation hub location model for a flood relief distribution network. *Computational Intelligence* 36, 3 (2020), 1320–1347.

[16] Marcone JF Souza, Igor Machado Coelho, Sabir Ribas, Haroldo G Santos, and Luiz Henrique de Campos Merschmann. 2010. A hybrid heuristic algorithm for the open-pit-mining operational planning problem. *European Journal of Operational Research* 207, 2 (2010), 1041–1051.

Uma meta-heurística adaptativa aplicada ao problema de roteamento de veículos capacitados e variantes

Angelina Gomes Santos
angelina.gomes@unifesp.br
Instituto de Ciência e Tecnologia
Universidade Federal de São Paulo
São José dos Campos, SP, Brasil

Antônio Augusto Chaves
antonio.chaves@unifesp.br
Instituto de Ciência e Tecnologia
Universidade Federal de São Paulo
São José dos Campos, SP, Brasil

Cleder Marcos Schenekemberg
cledercms@gmail.com
Instituto de Ciência e Tecnologia
Universidade Federal de São Paulo
São José dos Campos, SP, Brasil

RESUMO

Este trabalho busca resolver o Problema de Roteamento de Veículos Capacitados (CVRP, do inglês *Capacitated Vehicle Routing Problem*), que consiste em determinar rotas ótimas de entrega para uma frota de veículos a partir de um depósito, visitando cada cliente exatamente uma vez e respeitando a capacidade dos veículos, e algumas variantes utilizando a meta-heurística híbrida BRKGA-QL. Os dados utilizados foram extraído do Loggi *Benchmark for Urban Deliveries* e os tamanhos das instâncias variam de 7 mil a 32 mil entregas. Os métodos exatos podem ser aplicados para resolver pequenas instâncias do CVRP, porém para um problema com a dimensão abordada neste trabalho as meta-heurísticas podem ser a melhor opção. Um método adaptativo, BRKGA-QL, e um método exato baseado no algoritmo *Branch&Cut* (B&C) foram implementados e avaliados em um pequeno conjunto de dados obtido a partir do conjunto original da Loggi. Com isso, observou-se que o B&C foi capaz de resolver apenas instâncias pequenas enquanto o BRKGA-QL encontrou boas soluções para todas as instâncias analisadas.

CONCEITOS CCS

• **Computing methodologies → Heuristic function construction**.

PALAVRAS-CHAVE

Problema de Roteamento de Veículos Capacitados. BRKGA-QL. Q-Learning. Algoritmo Branch&Cut

Formato de Referência ACM:
Angelina Gomes Santos, Antônio Augusto Chaves, and Cleder Marcos Schenekemberg. 2021. Uma meta-heurística adaptativa aplicada ao problema de roteamento de veículos capacitados e variantes. In *Workshop PBP-Loggi 2021: Resultados do Programa de Bolsas de Pesquisa (PBP-Loggi)*. 6 pages.

1 INTRODUÇÃO

Alguns problemas de otimização encontrados na vida real podem ser melhor resolvidos com algoritmos aproximados, como meta-heurísticas, do que com métodos exatos, principalmente quando se tratam de problemas de otimização combinatória de grande escala, que consistem em maximizar ou minimizar uma função por meio da melhor combinação das variáveis do problema que, neste caso, assumem valores discretos (inteiros) e, mais especificamente, do Problema de Roteamento de Veículos (VRP - do inglês *Vehicle Routing Problem*).

O VRP foi introduzido por Dantzig e Ramser com o nome de *Truck Dispatching Problem* [5] e anos depois Clarke e Wright [3] definiram a generalização deste problema, que é um dos problemas mais estudados em Pesquisa Operacional e apresenta diversas variantes ao incorporar detalhes encontrados nos problemas do dia-a-dia da ciência, engenharia, matemática e afins.

Além de ser um problema de otimização combinatória, o VRP pertence à classe NP-difícil, possui diversas variações e pode ser aplicado para determinar estratégias de redução de custos operacionais em redes de distribuição, principalmente ao considerar as capacidades de cada veículo, o que leva à primeira variante: Problema do Roteamento de Veículos Capacitados (CVRP, do inglês *Capacitated Vehicle Routing Problem*), que consiste em determinar rotas ótimas de entrega para uma frota de veículos a partir de um depósito, visitando cada cliente exatamente uma vez e respeitando a capacidade dos veículos.

Muitas requisições adicionais e restrições operacionais são impostas à construção da rota em aplicações práticas do VRP. Por exemplo, o serviço pode incluir tanto entrega quanto coletas, a carga em cada rota não pode exceder uma dada capacidade do veículo, o tamanho das rotas pode ter que ser inferior a um limite pré-definido, o serviço pode incluir janela de tempo, a frota de veículos pode ser heterogênea, os clientes podem ter relações pré-existentes, as demandas podem não ser completamente conhecidas de início, o serviço de um cliente pode ser dividido em diferentes veículos e algumas características, como demandas e tempo de entrega, podem variar dinamicamente. [16]

Visto que o problema possui diversas variantes, é necessário o desenvolvimento de métodos flexíveis e que exijam mínimo esforço de ajuste de parâmetros e adaptação para diferentes variações por parte dos desenvolvedores. E uma meta-heurística capaz de ser utilizada para resolver diferentes problemas de otimização combinatória é o *Biased Random-Key Genetic Algorithm* (BRKGA) [8], um algoritmo evolutivo que utiliza chaves aleatórias para representar a solução e algumas adaptações podem facilitar ainda mais sua utilização, como a incorporação do algoritmo *Q-learning* de aprendizagem por reforço para realizar a configuração dos parâmetros, gerando o BRKGA-QL [2]. Dessa forma, o presente trabalho tem o intuito de aplicar o BRKGA-QL para resolver o CVRP e uma variante incorporando coleta de prêmios (*Prize-collecting VRP*, PCVRP), além de apresentar modelagens matemáticas para os problemas abordados.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Workshop PBP-Loggi, 08 de dezembro de 2021, Brasil

Os dados utilizados foram extraídos do Loggi *Benchmark for Urban Deliveries* (BUD) que contém conjuntos de dados e *scripts* de *benchmark* para problemas de grande escala usando como teste algumas das maiores cidades do Brasil. Os dados são sintetizados completamente a partir de fontes públicas e representativos dos desafios enfrentados pela Loggi nas entregas do mundo real. São fornecidas 90 instâncias de treino e 30 de validação para o problema no Rio de Janeiro (RJ), Brasília (DF) e Belém (PA). Os tamanhos das instâncias variam de 7 mil a 32 mil entregas [10].

2 OBJETIVOS

2.1 Objetivo Geral

O presente trabalho tem o intuito de aplicar o BRKGA-QL para resolver o CVRP e uma de suas variantes, além de apresentar modelagens matemáticas para os problemas abordados, assim elaborando uma ferramenta ágil que ajude os usuários a entenderem como lidar com problemas difíceis e uma grande massa de dados.

2.2 Objetivos Específicos

- Apresentar formulações matemáticas para os problemas e resolvê-los usando o algoritmo B&C;
- Desenvolver um *framework* usando o BRKGA-QL com implementação de *decoders* e heurísticas específicas para resolver o CVRP e a variante com coleta de prêmios;
- Realizar testes computacionais sobre o método adaptativo proposto.

3 REVISÃO BIBLIOGRÁFICA

Há mais de 60 anos, Dantzig e Ramser introduziram o problema, na época denominado *Truck Dispatching Problem* [5], ao modelar o atendimento de postos de gasolina a partir de um depósito central para minimizar o caminho percorrido por uma frota de caminhões. Anos depois, Clarke e Wright definiram um dos problemas mais estudados na Pesquisa Operacional, o *Vehicle Routing Problem* [3], uma generalização do problema de Dantzig e Ramser. Este problema consiste em determinar rotas de menor custo a partir de um depósito para atender um conjunto de clientes.

O VRP é um problema amplamente estudado, em 2009 Eksioglu, Vural e Reisman [6] mostraram que 1021 artigos, cujo assunto principal era o VRP, foram publicados entre 1959 e 2008 e também que os estudos do VRP tem crescido quase que exponencialmente numa taxa superior a 6% ao ano.

Como já mencionado, muitas requisições e restrições podem ser adicionadas ao VRP dando origem a algumas variantes, como é o caso do problema em foco que leva em conta o fato dos veículos possuírem uma capacidade. Outra variação interessante é o Problema de Roteamento de Veículos com Coleta de Prêmios (PCVRP do inglês *Prize-Collecting Vehicle Routing Problem*) [15], em que um prêmio é coletado ao visitar um cliente e multas são incorridas para cada cliente não visitado.

Sobre os métodos utilizados neste projeto, o *Biased Random-Key Genetic Algorithm* (BRKGA) [8] é um algoritmo evolutivo que usa o conceito de chaves aleatórias para representar uma solução. A ideia deste método é ser eficiente e robusto para resolver diferentes problemas de otimização, baseando a busca pelo espaço de soluções de chaves aleatórias e fazendo o mapeamento para o espaço de solução do problema por meio de uma função determinística, chamada *decoder*. Já o *Q-Learning* (QL) [17], que é considerado um dos algoritmos mais utilizados em Aprendizagem por Reforço, tem o objetivo de escolher uma ação adequada com base na experiência e obter uma recompensa ou penalidade para cada ação realizada.

Uma variação da meta-heurística BRKGA foi proposta por Chaves, Gonçalves e Lorena [1] com o intuito de gerar um método com controle *on-line* de parâmetros, exigindo que o desenvolvedor definisse apenas uma função *decoder*, responsável por calcular a função objetivo e o tratamento de restrições. O método denominado A-BRKGA, no entanto, teve muitas decisões definidas no início e mantidas durante todo o processo de busca. Desta forma, acoplando o algoritmo *Q-Learning* para controlar os parâmetros do BRKGA, Chaves e Lorena [2] propuseram o BRKGA-QL, nunca avaliado no CVRP, ele tem os parâmetros tamanho da população (p), tamanho das partições elite (p_e) e mutante (p_m), bem como a probabilidade de herdar a chave do pai elite (ρ_e), definidos pelo algoritmo QL durante o processo evolutivo, ao invés de usar um valor fixo e pré-definido de parâmetro.

4 MODELO MATEMÁTICO

De acordo com Laporte [9], o CVRP clássico é definido sobre um grafo direcionado $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, onde $\mathcal{V} = \{0, 1, ..., n\}$ representa o conjunto de vértices e $\mathcal{A} = \{(i, j) : i, j \in \mathcal{V}, i \neq j\}$ o conjunto de arcos. Além disso, um custo não negativo, c_{ij} é associado a cada arco $(i,\ j) \in \mathcal{A}$. O vértice 0 indica o depósito, ponto inicial e final de cada rota, enquanto os índices de 1 até n indicam os clientes. A frota disponível dispõe de m veículos idênticos de capacidade Q e cada cliente i possui uma demanda não-nula de q_i unidades. O objetivo deste problema é planejar rotas de custo mínimo tal que todos os veículos efetuem apenas uma rota, a capacidade de cada veículo seja respeitada e cada cliente seja atendido exatamente uma vez.

Seguindo o modelo apresentado por Toth e Vigo [16], o CVRP pode ser formulado conforme apresentado nas restrições de (1) a (7), onde c_{ij} denota o custo do arco $(i,\ j)$ e a variável binária x_{ij} que assume valor igual a 1 se o arco $(i,\ j)$ pertence à solução, caso contrário $x_{ij} = 0$.

$$min \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} c_{ij} x_{ij}\ ,\ i \neq j \tag{1}$$

$$s.a. \quad \sum_{i \in \mathcal{V}} x_{ij} = 1 \quad \forall\ j \in \mathcal{V} \backslash \{0\},\ i \neq j \tag{2}$$

$$\sum_{j \in \mathcal{V}} x_{ij} = 1 \quad \forall\ i \in \mathcal{V} \backslash \{0\},\ i \neq j \tag{3}$$

$$\sum_{i \in \mathcal{V}} x_{i0} \leq m \quad ,\ i \neq j \tag{4}$$

$$\sum_{j \in \mathcal{V}} x_{0j} \leq m \quad ,\ i \neq j \tag{5}$$

$$\sum_{i \notin \mathcal{S}} \sum_{j \in \mathcal{S}} x_{ij} \geq \gamma(\mathcal{S}) \quad \forall\ \mathcal{S} \subseteq \mathcal{V} \backslash \{0\},\ \mathcal{S} \neq \emptyset,\ i \neq j \tag{6}$$

$$x_{ij} \in \{0, 1\} \quad \forall\ i, j \in \mathcal{V},\ i \neq j. \tag{7}$$

A equação (1) representa a função objetivo e é dada pela soma dos custos de todos os arcos que pertencem à solução do problema.

As restrições (2) e (3) garantem que apenas um arco liga i a j, com exceção do depósito. Além disso, (4) e (5) garantem que cada solução possua no máximo m rotas. Por fim, as restrições (6) definem os corte de capacidade, que estipulam que cada corte $(\mathcal{V}\backslash\mathcal{S};\mathcal{S})$ definido por um conjunto de vértices $\mathcal{S}$ é cruzado por um número de arcos não menor que o mínimo de veículos necessários para servir o conjunto $\mathcal{S}$, $\gamma(\mathcal{S})$, impondo a conectividade da solução e os requisitos de capacidade do veículo.

Note que, (6) é similar a (8), restrições generalizadas de eliminação de *subtour* (GSEC), obtida pela transformação da primeira por meio das restrições de grau (2 - 5).

$$\sum_{i\in\mathcal{S}}\sum_{j\in\mathcal{S}} x_{ij} \leq |\mathcal{S}| - \gamma(\mathcal{S}) \quad \forall\, \mathcal{S}\subseteq\mathcal{V}\backslash\{0\},\ \mathcal{S}\neq\emptyset,\ i\neq j \quad (8)$$

O modelo apresentado pode ser totalmente gerado apenas para instâncias de tamanho muito pequeno já que, para problemas de tamanho realistas, o número de restrições de corte de capacidade (6) aumenta exponencialmente (na ordem de 2^n) e sua enumeração completa é impraticável. Para superar esta limitação, usa-se um algoritmo *branch-and-cut* (B&C), em que estas restrições são dinamicamente geradas e adicionadas ao longo do processo de busca, sempre que alguma restrição de corte de capacidade é violada.

Neste trabalho foi utilizado um modelo incompleto, sem as restrições (6), para ser resolvido com o B&C que funciona da seguinte forma. Inicialmente, todas as restrições (exceto 6) são geradas, a cada solução obtida verifica-se a existência de alguma rota com subrota ou que exceda a capacidade do veículo, isto é, se existe alguma restrição de corte de capacidade que é violada. Caso a rota de algum veículo seja violada sua desigualdade (6) correspondente é adiciona ao modelo, em tempo de execução, evitando que tal solução seja aceita. Este processo é repetido até que uma solução viável ou dominada seja alcançada, ou até que não haja mais cortes a serem adicionados.

E, adaptando o modelo anterior, o PCVRP pode ser formulado de acordo com as equações de (9) a (16), p_i indica a prioridade do cliente i e $y_i = 1$ se o cliente i pertencer à solução, caso contrário $y_i = 0$. Note que distância e prioridade são medidas distintas, sendo assim um peso α equilibra o somatório.

$$min\ \alpha\left[\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}} c_{ij}x_{ij}\right] + (1-\alpha)\left[\sum_{i=1}^{n} p_i y_i\right],\ i\neq j \quad (9)$$

$$s.a. \quad \sum_{i\in\mathcal{V}} x_{ij} = y_j \quad \forall\, j\in\mathcal{V}\backslash\{0\},\ i\neq j \quad (10)$$

$$\sum_{j\in\mathcal{V}} x_{ij} = y_i \quad \forall\, i\in\mathcal{V}\backslash\{0\},\ i\neq j \quad (11)$$

$$\sum_{i\in\mathcal{V}} x_{i0} \leq m \quad ,\ i\neq j \quad (12)$$

$$\sum_{j\in\mathcal{V}} x_{0j} \leq m \quad ,\ i\neq j \quad (13)$$

$$\sum_{i\notin\mathcal{S}}\sum_{j\in\mathcal{S}} x_{ij} \geq \gamma(\mathcal{S}) \quad \forall\, \mathcal{S}\subseteq\mathcal{V}\backslash\{0\},\ \mathcal{S}\neq\emptyset,\ i\neq j \quad (14)$$

$$x_{ij}\in\{0,1\} \quad \forall\, i,j\in\mathcal{V},\ i\neq j \quad (15)$$

$$y_i\in\{0,1\} \quad \forall\, i\in\mathcal{V}\backslash\{0\}. \quad (16)$$

Aqui a equação (9) representa a função objetivo e é dada pela soma dos custos de todos os arcos que pertencem à solução do problema e pela soma das prioridades de todos os clientes atendidos. As equações (10) e (11) garantem que apenas um arco liga i a j, com exceção do depósito, e também que i e j de fato fazem parte da solução. As equações de (12) a (16) são respectivamente equivalentes as equações de (4) a (7).

5 MÉTODOS

5.1 BRKGA

Biased Random-Key Genetic Algorithm (BRKGA) [8] é um algoritmo evolutivo que representa uma solução por meio de chaves aleatórias e utiliza uma função determinística chamada *decoder* para fazer o mapeamento para o espaço de solução do problema.

Tamanho da população (p), tamanho das partições elite (p_e) e mutante (p_m) e probabilidade de herdar a chave do pai elite (ρ_e) são os parâmetros do BRKGA e o processo evolutivo segue: inicialmente a população é dividida em dois grupos (Elite e Não Elite), a partição Elite é copiada com os melhores $p * p_e$ cromossomos para a próxima população, então são criados $p * p_m$ novos vetores de chaves aleatórias (mutantes) com o mesmo processo usado para criar a população inicial e gerados filhos com o *Parametric Uniform Crossover* [14] para completar a próxima população. Cada cromossomo novo (filhos e mutantes) é avaliado pelo *decoder* e, após a avaliação, a nova população é ordenada outra vez.

O algoritmo do BRKGA é retratado na figura 1 onde é possível ver a divisão entre problema-dependente e problema-independente, sendo que a parte independente não tem nenhum conhecimento sobre o problema de otimização que está sendo resolvido, ao contrário da parte dependente, que é onde o *decoder* gera a solução do problema a partir das chaves aleatórias. Mais detalhes sobre o BRKGA podem ser lidos em [8].

Figure 1: Fluxograma do BRKGA. Adaptado do artigo Biased random-key genetic algorithms for combinatorial optimization de Gonçalves e Resende [8].

Como toda meta-heurística, o BRKGA exige que seja realizada a configuração de seus parâmetros para cada problema, mas cada etapa do processo evolutivo pode ter uma configuração adequada diferente, o que dificulta mais a tarefa. Por isso, neste trabalho, é utilizado o BRKGA-QL, que incorpora o algoritmo *Q-learning* [17] de aprendizagem por reforço para definir os parâmetros do BRKGA ao longo do processo evolutivo.

5.2 BRKGA-QL

Cada parâmetro do BRKGA-QL é representado por um estado (s) e as ações (a) que podem ser realizadas são os valores que os parâmetros do algoritmo podem assumir. Além disso, os parâmetros do BRKGA e do QL são definidos a partir da tabela Q, aplicando a política ϵ-*Greedy* em cada geração do BRKGA e, após cada processo evolutivo, é utilizada a equação (17), do QL, para atualizar os valores Q das ações escolhidas para todos os parâmetros.

$$Q(s,a) = Q(s,a) + lf\left[R + df * \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s,a)\right] \quad (17)$$

sendo que, s é o estado atual, a é a ação realizada no estado s, R é o sinal de reforço (recompensa), s_{t+1} é o próximo estado, lf é o fator de aprendizagem, df é o fator de desconto do QL.

Os valores recomendados na literatura para os parâmetros do BRKGA [13] e do QL [12] foram utilizados para definir a lista de ações, com exceção do tamanho da população, que é definido pela sequência de Fibonacci $p = \{233, 377, 610, 987, 1597, 2584\}$, a fim de permitir que nenhum parâmetro precise ser manualmente configurado.

O BRKGA-QL inicializa definindo os valores da tabela Q como zero. A população inicial é gerada com p vetores de chaves aleatórias (cromossomos). Cada cromossomo possui $n+1$ genes gerados aleatoriamente no intervalo $[0, 1]$ com probabilidade uniforme. O valor de n pode ser definido pelo número de clientes no CVRP.

O *decoder* avalia a qualidade de cada cromossomo. No BRKGA-QL trabalha-se com a opção dos usuários definirem k *decoders* diferentes. Para isso, uma nova chave aleatória na posição $n + 1$ é utilizada para definir qual *decoder* é aplicado em cada cromossomo.

O processo evolutivo do BRKGA-QL é o mesmo do BRKGA clássico, divide-se a população em dois grupos (Elite e Não Elite), copia-se a partição Elite com os melhores $p * p_e$ cromossomos para a próxima população, criam-se $p * p_m$ novos vetores de chaves aleatórias (mutantes) com o mesmo processo usado para criar a população inicial e geram-se filhos com o *Parametric Uniform Crossover* [14] para completar a próxima população. Cada cromossomo novo (filhos e mutantes) é avaliado pelo *decoder* definido em sua chave aleatória $n + 1$. Após a avaliação, a nova população é novamente ordenada.

Se a configuração de parâmetros selecionada encontrar um novo melhor cromossomo na população, a recompensa (R) é definida com valor 1 e a função $Q(s, a)$ será atualizada para todos os pares estado/ação (parâmetro/valor) pela equação 17.

Com um tamanho de população adaptativo, a necessidade de definir um certo número de gerações torna-se menos importante. Portanto, este parâmetro pode ser removido e, em vez disso, utiliza-se um tempo máximo de execução, run_{time} (em segundos) como critério de parada do BRKGA-QL.

5.3 Decoders

Foram desenvolvidos quatro *decoders* sendo que todos possuem o primeiro passo em comum, que é gerar uma solução inicial, ou seja, associar cada chave aleatória rk (solução do BRKGA) a um vértice (cliente) do problema específico. Depois disso, um dos *decoders* é escolhido aleatoriamente para dar continuidade ao processo de decodificação do vetor de rk.

Como já mencionado na subseção 5.2, o vetor de chaves aleatórias (que representa um cromossomo para o algoritmo genético) possui $n + 1$ posições com valores no intervalo $[0, 1]$ (genes), sendo que n é dado pelo número de clientes no CVRP.

O *decoder 1* é o mais simples e utiliza apenas um método de ordenação, o procedimento completo realizado é o seguinte: primeiramente, o vetor de solução de chaves aleatórias é ordenado e, então, cada ponto é alocado nas rotas dos veículos respeitando suas capacidades unitárias. Quando o ponto a ser inserido possui uma demanda superior à capacidade disponível no veículo corrente, um novo veículo é aberto e as inserções passam a ocorrer nele. Por isso, utilizando o *decoder 1*, não necessariamente será atingida a capacidade máxima dos veículos, de acordo com sua relação com a demanda apresentada.

Figure 2: Fluxograma do *Decoder 1*.

Os demais *decoders* também iniciam ordenando o vetor solução e seguem essa ordem, de algum modo, para designar os clientes aos veículos. O *decoder 2* diferencia-se do *1* por utilizar a ideia de inserção mais barata, ou seja, sempre que um novo ponto vai ser inserido na rota do veículo aberto, verifica-se qual a melhor posição para inserir este novo cliente dentre as posições já existes na rota.

Figure 3: Fluxograma do *Decoder 2*.

Exemplificando o funcionamento dos *decoders* acima mencionados, sejam $\{a, b\}$ uma rota e c o próximo ponto a ser inserido na rota. O *decoder 1* insere c ao final da rota já existente adicionando o custo do arco (b, c) à função objetivo. Já o *decoder 2* percorre toda a rota existente verificando em qual posição o custo adicionado à função objetivo será menor, ou seja, verifica para cada posição da rota se $c(i, j) > c(i, k) + c(k, j)$, assim, para este exemplo, o ponto c poderia ser inserido antes de a, entre a e b ou depois de b.

Para implementar o *decoder 3* foi utilizado o método 2-opt [4, 7], que compara todas as combinações possíveis de troca de caminhos em uma dada rota. Assim, é necessário alocar os clientes em rotas pré-determinadas para que o 2-opt possa ser aplicado a elas. Então,

após ordenar o vetor solução (clientes), eles são divididos entre m veículos, onde m é o número mínimo de veículos necessários para atender a demanda total dos clientes. Em seguida, respeitando a ordenação dos clientes, é verificado se a capacidade dos veículos não foi extrapolada. Nos casos em que isso ocorre, novos veículos (tantos quantos forem necessários) são abertos para atender os clientes excedentes.

Com todos os clientes alocados, sem ultrapassar a capacidade dos veículos, o 2-opt é aplicado em cada rota. Sejam, i e j dois pontos da rota tal que $j \neq i$ e is e js os sucessores de i e j, respectivamente, compara-se a distância total da rota atual com a distância da rota obtida a partir da remoção dos arcos (i, is) e (j, js) e inserção dos arcos (i, j) e (is, js) e sempre que uma solução melhor (de menor distância) é encontrada, a alteração dos arcos é realizada. O processo se repete por $n * (n - 2)$ comparações, já que o vetor de solução é percorrido uma vez realizando $n - 2$ comparações a cada posição.

Por fim, o *decoder 4* é bem parecido com o *decoder 3*, sendo que a única diferença é que não é aplicado o método 2-opt e sim o *swap* (troca). Neste método, é realizada a troca da posição de dois clientes i e j na rota sempre que esta troca resulta em uma solução melhor que a encontrada até então. Sejam, ip, is, jp e js o predecessor e sucessor de i e o predecessor e sucessor de j, respectivamente, quando i e j trocam de lugar, os arcos (ip, i), (i, is), (jp, j) e (j, js) são removidos e os arcos (ip, j), (j, is), (jp, i) e (i, js) são inseridos na rota.

5.4 Buscas Locais

A busca local é mais uma etapa para melhorar as soluções encontradas, ela é aplicada sempre que ocorre uma melhora na solução (exploração) ou quando o melhor cromossomo não obtém melhora por algumas gerações (estagnação).

Foram utilizadas quatro estratégias de busca local, sendo duas intra rota e duas inter rota. As estratégias intra rota são aquelas em que as operações são realizadas apenas entre elementos de uma mesma rota, por exemplo, remover um elemento da posição i e inseri-lo em $i + 2$. Já as estratégias inter rotas são aquelas em que as operações ocorrem entre rotas distintas, como a remoção de um nó i da rota $r1$ e inserção deste nó na rota $r2$. No caso das estratégias inter rota, é necessário levar em conta a distância total de todas as rotas envolvidas.

Os métodos implementados foram o *swap* e o *shift*, cada um nas versões intra e inter rota. O *swap* apresentado na subseção 5.3 é a versão intra rota, uma vez que i e j encontram-se na mesma rota. Para a versão inter rotas basta tomar i em uma rota $r1$ e j em uma rota diferente $r2$, deste modo a troca será realizada entre elementos de rotas distintas, por isso inter rotas.

No *shift* apenas um elemento i muda de lugar passando a ocupar uma nova posição a frente de sua posição atual e as trocas são realizadas sempre que resultam em uma solução melhor. Note que, ao remover o nó i de sua posição original e inseri-lo em outra posição as ligações com seu predecessor ip e seu sucessor is são quebradas e também as ligações entre um nó j e seu sucessor js, entre os quais i será inserido. Ou seja, os arcos (ip, i), (i, is) e (j, js) são removidos e os arcos (ip, is), (j, i) e (j, js) são inseridos.

6 RESULTADOS PRELIMINARES

Buscando validar o que foi desenvolvido, foram realizados testes utilizando um algoritmo *Branch&Cut* (B&C) para obter a solução exata de um conjunto de instâncias menores, com 30, 60, 90 e 120 pontos, que foram extraídas dos dados fornecidos pela Loggi. Assim, foi possível comparar os resultados dos testes realizados com o BRKGA-QL e estas mesmas instâncias.

A partir dos dados fornecidos no Loggi BUD, mais especificamente na pasta cvrp-instances-1.0/train, foram geradas 44 novas instâncias com 30, 60, 90 e 120 pontos, sendo 11 de cada tamanho e de modo que tenha apenas uma instância de cada hub com cada tamanho. Ou seja, foram utilizados os 4 primeiros arquivos presentes em cada uma das 11 pastas, extraindo os pontos sempre do início do arquivo, sendo 30 do arquivo 1, 60 do arquivo 2, 90 do arquivo 3 e 120 do arquivo 4.

Então, o B&C foi executado com limite máximo de 2 horas, 8 *threads* e o Gurobi 9.1 [11] foi utilizado para resolver cada instância, enquanto o BRKGA-QL executou cada instância 10 vezes por n segundos, onde n é o número de pontos. O B&C não foi capaz de resolver uma instância de 60 pontos, cinco de 90 e oito de 120. A Tabela 1 apresenta a solução ótima obtida com o B&C, o tempo de execução do B&C, o menor valor encontrado com o BRKGA-QL, o gap (18) em relação a melhor solução encontrada pelo BRKGA-QL , a solução média obtida pelo BRKGA-QL, dada por $M = \frac{\sum_{i=1}^{10} obtido_i}{10}$, e o desvio da solução média (19) para as 30 instâncias que o B&C conseguiu resolver.

$$GAP = \frac{obtido - ótimo}{ótimo} * 100 \tag{18}$$

$$Des = \frac{M}{ótimo} * 100 \tag{19}$$

Por fim, a Tabela 2 apresenta o *lower bound* e *upper bound* encontrados pelo B&C, o tempo de execução do B&C e o menor valor e a solução média encontrados pelo BRKGA-QL para as demais instâncias.

7 CONSIDERAÇÕES PARCIAIS

Neste trabalho foram estudados métodos de otimização para propor soluções para o problema do roteamento de veículos capacitados, tendo como foco o BRKGA-QL. Além dos decoders específicos desenvolvidos para o método adaptativo utilizado e aplicação de métodos de busca local, foi implementado também um algoritmo *branch-and-cut* utilizado para validação dos resultados obtidos com a meta-heurística.

De modo geral, o BRKGA-QL apresentou um bom resultado com GAPs abaixo de 10%, com exceção de três *outliers*. Além disso, para as instâncias que o B&C não foi capaz de resolver dentro do limite de tempo estipulado, o BRKGA-QL apresentou melhores resultados comparados ao *upper bound* do B&C em aproximadamente 64, 29% dos casos.

8 TRABALHO EM ANDAMENTO

Tendo o método validado, o próximo passo é aplicá-lo em instâncias maiores e realizar analises estatísticas sobre os resultados obtidos. Além disso, o PCVRP tem sido estudado para ser aplicado em uma

Table 1: Resultados das instâncias com solução ótima.

Instância	B&C	T(s)	BRKGA-QL	Gap	Média	Des
30-0-df	152,39	0,11	153,06	0,44	153,06	0,44
30-0-pa	77,46	0,12	79,43	2,55	79,43	2,55
30-0-rj	10,71	0,09	10,71	0,00	10,71	0,00
30-1-df	11,12	0,16	11,12	0,00	11,12	0,00
30-1-pa	27,69	0,05	27,69	0,00	27,69	0,00
30-1-rj	11,70	0,04	11,70	0,00	11,70	0,00
30-2-df	31,07	0,08	31,07	0,00	31,07	0,00
30-2-rj	36,15	0,05	36,15	0,00	36,15	0,00
30-3-rj	6,20	0,16	6,20	0,00	6,20	0,00
30-4-rj	4,93	0,06	4,93	0,00	4,93	0,00
30-5-rj	6,92	0,03	6,92	0,00	6,92	0,00
60-0-df	140,89	42,50	140,89	0,00	142,01	0,80
60-0-pa	83,63	3,77	84,97	1,60	85,16	1,83
60-0-rj	15,29	2,29	15,38	0,61	15,65	2,37
60-1-df	30,61	2,51	30,71	0,31	30,78	0,56
60-1-pa	73,51	0,65	73,53	0,03	73,56	0,06
60-1-rj	15,11	0,56	15,34	1,54	15,65	3,60
60-2-df	66,96	1,12	67,30	0,50	67,35	0,58
60-3-rj	10,30	5,09	10,41	1,08	10,45	1,47
60-4-rj	8,59	558,89	8,61	0,28	8,65	0,77
60-5-rj	11,10	11,65	11,15	0,47	11,19	0,80
90-0-pa	95,05	66,62	96,98	2,03	98,71	3,85
90-0-rj	19,04	96,00	20,53	7,82	21,20	11,31
90-1-df	40,41	37,42	48,75	20,62	48,91	21,03
90-1-pa	87,81	2,06	96,12	9,47	106,20	20,95
90-1-rj	19,65	42,50	21,46	9,23	21,96	11,77
90-2-df	106,64	190,91	113,21	6,16	133,34	25,04
120-0-rj	22,32	1.864,15	25,82	15,65	28,00	25,41
120-1-rj	23,45	3.314,82	27,72	18,20	28,90	23,26
120-2-df	147,23	4.088,28	155,78	5,81	182,55	23,99

Table 2: Resultados das instâncias sem solução ótima.

Instância	B&C - LB	B&C - UB	T(s)	BRKGA-QL	Média
60-2-rj	62,94	63,11	7200,17	63,30	63,61
90-0-df	251,73	626,63	7203,70	291,23	302,03
90-2-rj	85,85	97,80	7202,14	90,96	91,66
90-3-rj	14,63	15,09	7202,09	14,99	15,13
90-4-rj	11,22	65,28	7203,97	11,90	11,98
90-5-rj	15,66	45,23	7202,96	16,74	16,83
120-0-df	225,02	254,88	7202,72	263,13	278,76
120-0-pa	133,40	285,12	7201,48	142,17	144,70
120-1-df	42,71	42,77	7200,79	60,25	60,64
120-1-pa	101,78	103,99	7200,20	117,09	127,54
120-2-rj	105,81	119,28	7203,55	125,44	135,96
120-3-rj	18,25	24,43	7203,93	19,38	19,52
120-4-rj	22,26	36,12	7202,58	24,65	24,94
120-5-rj	17,60	26,80	7202,55	20,44	20,80

situação que considera que os clientes, além das demandas, possuem também prioridades.

A prioridade é indicada por um número inteiro no intervalo $[1, 5]$ que indica em qual dia a entrega deve ser realizada, ou seja, o cliente com prioridade 1 deve receber seu pedido no dia 1 (dia atual) e o cliente com prioridade 5 deve receber seu pedido até o dia 5. Então, uma penalidade é acrescida à função objetivo levando em consideração a prioridade de cada cliente. Desse modo, a nova função objetivo passa a buscar minimizar não apenas a distância percorrida, mas também a penalização das prioridades.

AGRADECIMENTOS

Este trabalho foi realizado com o apoio da Loggi (www.loggi.com), por meio do Programa de Bolsas de Pesquisa Loggi (PBP-LOGGI).

REFERENCES

[1] Antonio Augusto Chaves, Jose Fernando Goncalves, e Luiz Antonio Nogueira Lorena. 2018. Adaptive biased random-key genetic algorithm with local search for the capacitated centered clustering problem. *Computers & Industrial Engineering* 124 (2018), 331–346.

[2] Antônio Augusto Chaves e Luiz Henrique Nogueira Lorena. 2021. An adaptive and near parameter-free brkga using q-learning method. In *2021 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2331–2338.

[3] Geoff Clarke e John W. Wright. 1964. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research* 12, 4 (Aug. 1964), 519—-643. https://doi.org/10.1287/opre.12.4.568

[4] Georges A Croes. 1958. A method for solving traveling-salesman problems. *Operations research* 6, 6 (1958), 791–812.

[5] George B. Dantzig e John H. Ramser. 1959. The truck dispatching problem. *Management Science* 6, 1 (Oct. 1959), 80–91. https://doi.org/10.1287/mnsc.6.1.80

[6] Burak Eksioglu, Arif Volkan Vural, e Arnold Reisman. 2009. The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering* 57, 4 (2009), 1472–1483.

[7] Merrill M Flood. 1956. The traveling-salesman problem. *Operations research* 4, 1 (1956), 61–75.

[8] José Fernando Gonçalves e Mauricio GC Resende. 2011. Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics* 17, 5 (2011), 487–525.

[9] Gilbert Laporte. 2009. Fifty years of vehicle routing. *Transportation science* 43, 4 (2009), 408–416.

[10] Loggi. 2021. loggiBUD: Loggi Benchmark for Urban Deliveries. https://github.com/loggi/loggibud.

[11] Gurobi Optimization-LLC. 2021. Gurobi 9.1. https://www.gurobi.com/.

[12] Hande Öztop, Mehmet Fatih Tasgetiren, Levent Kandiller, e Quan-Ke Pan. 2020. A novel general variable neighborhood search through q-learning for no-idle flowshop scheduling. In *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1–8.

[13] Hari Prasetyo, Gusti Fauza, Yousef Amer, e Sang-Heon Lee. 2015. Survey on applications of biased-random key genetic algorithms for solving optimization problems. In *2015 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*. IEEE, 863–870.

[14] William M. Spears e Kenneth A. De Jong. 1991. On the Virtues of Parameterized Uniform Crossover. *Proc. of the Fourth International Conference on Genetic Algorithms* (1991), 230–236.

[15] Andreas Stenger, Michael Schneider, e Dominik Goeke. 2013. The prize-collecting vehicle routing problem with single and multiple depots and non-linear cost. *EURO Journal on Transportation and Logistics* 2, 1-2 (2013), 57–87.

[16] Paolo Toth e Daniele Vigo. 2002. Models, relaxations and exact approaches for the capacitated vehicle routing problem. *Discrete Applied Mathematics* 123, 1-3 (2002), 487–512.

[17] Christopher John Cornish Hellaby Watkins. 1989. Learning from delayed rewards. (1989).

Um problema de roteamento de veículos com frota heterogênea

Modelo de otimização e heurísticas

Lee Jing Xuan
Instituto de Matemática
Universidade Federal de Mato Grosso do Sul
Campo Grande, MS, Brasil
leejingxuan2000@gmail.com

Willy Alves de Oliveira
Instituto de Matemática
Universidade Federal de Mato Grosso do Sul
Campo Grande, MS, Brasil
willy.oliveira@ufms.br

RESUMO

O Problema de Roteamento de Veículos (VRP) é um problema de otimização amplamente estudado na literatura. Existem diversas variações, dentre as quais, o VRP heterogêneo com custos fixos e custos dependentes da rota (Heterogeneous VRP with Fixed Costs and Vehicle Dependent Routing Costs HVRPFD). A finalidade deste trabalho é estudar uma formulação matemática para essa variante encontrada na literatura, bem como a construção de uma heurística construtiva gulosa e de procedimentos heurísticos de refinamento, visando a obtenção de boas soluções factíveis em tempo computacional aceitável.

Palavras-chaves: VRP; HVRPFD; formulação matemática; heurísticas

1. Os objetivos da pesquisa

Segundo o estudo da literatura, pode-se perceber que o problema de HVRPFD apresenta alto grau de complexidade na sua resolução, uma vez que apresenta mais aspectos incluídos, aumenta exponencialmente o número de possibilidades nas decisões a serem tomadas. Essa quantidade enorme de dados, escolhas e resultados afetam negativamente a obtenção de boas soluções factíveis em tempo computacional aceitável. Diante disso, buscando atenuar as dificuldades anteriores, os objetivos da pesquisa são:

1. Estudar modelos matemáticos para o problema;

2. Estudar e implementar um modelo matemático do problema em linguagem computacional;

3. Desenvolver procedimentos heurísticos para construção de soluções factíveis em baixo tempo computacional;

4. Investigar procedimentos de melhoria baseados em busca local para aprimorar as soluções factíveis encontradas.

2. As atividades desenvolvidas

2.1 A compreensão de um modelo matemático para o HVRPFD

Antes de detalhes do problema, é básico entender primeiro as modelagens matemáticas, desde as ideias cotidianas de acelerar um determinado processo até otimizações com ferramentas computacionais. Para um alicerce do assunto a ser tratado, sugere-se as abordagens em [1].

Os problemas de roteamentos de veículos podem ser entendidos partindo do problema de caixeiro viajantes (*Travelling Salesman Problem* TSP), outro problema clássico da literatura. Suponha que se deseja realizar entregas para uma quantidade de n clientes com apenas um veículo, onde a capacidade do veículo não é limitada. Neste caso, contamos com a saída deste único veículo do depósito para os n clientes e o seu retorno ao depósito no final de todas entregas. O objetivo deste problema seria encontrar o caminho mais curto ou de menor custo, passando por todos os clientes uma única vez e terminando a viagem no ponto de partida. Os detalhes sobre TSP, seja formulação matemática e suas possíveis heurísticas, são facilmente encontrados na literatura, como, por exemplo, em [2]. Para detalhes com enfoque na história, algoritmo e aplicações do problema, pode ser encontrado, por exemplo, em [3].

Com base no problema do TSP, pode-se entender o HVRPFD fazendo algumas alterações. A quantidade de veículos é conhecida, mas nem sempre é única, além disso, a capacidade de cada veículo passa a ser relevante, e então, cada tipo de veículo com capacidade distinta terá seu custo fixo. Dessa forma, observe que houve uma grande amplificação nas possibilidades dos veículos que sairão do depósito, e das rotas que cada um dos veículos irá percorrer. O objetivo desse problema será encontrar os veículos que sairão do depósito e suas respectivas rotas, visando minimizar o custo total, ou seja, o custo fixo de cada veículo somado com todo custo de cada rota percorrido.

O Problema de Roteamento de Veículos Heterogêneo com Custos fixos e Custos dependentes da rota possui três características específicas:

(i) A frota de veículos é composta por número limitado de veículos para cada tipo
(ii) São considerados os custos fixos dos veículos
(iii) Os custos de roteamento são dependentes do veículo

A primeira característica é a quantidade limitada de veículos de cada tipo. Na resolução do problema deverá considerar essas disponibilidades para encontrar as rotas de solução.

A segunda característica é que cada tipo de veículo possui um custo fixo associado. Este é um novo critério avaliativo para a escolha das rotas com menor custo. Por exemplo, suponha que para dois veículos, tenha o mesmo custo de roteamento num trajeto, e assim,

gera a dúvida por qual veículo optar para fazer o trabalho. No entanto, sabendo que o custo total é o custo fixo somado com custo do trajeto e que um veículo tem custo fixo menor que o outro, naturalmente optaremos pelo veículo com menos custo total.

A terceira característica explicita que pode haver diferenças de custo de roteamento numa mesma rota quando percorrido por veículos de tipos diferentes. Pode-se relacionar essa situação com o pagamento do pedágio, para cada categoria é pagado um valor diferente, e assim, numa rota que contém pedágio, dois veículos podem exigir custos distintos.

2.2 Um modelo matemático para o HVRPFD

2.2.1 Notação e Variáveis

Primeiramente, um grafo direcionado $G(V, A)$ é atribuído, $V = \{0,1, \dots, n\}$ é o conjunto com n+1 nós e A é o conjunto de arcos. Vale salientar que o grafo sendo direcionado, as arestas ou arcos possuem um sentido ou direção relevante. O nó 0 representa o depósito e o conjunto $V' = V \backslash \{0\}$ representa os clientes. Todo cliente $i \in V'$ requer uma quantidade q_i de fornecimento do depósito (assume-se $q_0 = 0$). Uma frota heterogênea de veículos está estacionada no depósito para atender os clientes, a qual é composta por m diferentes tipos de veículos, e assim, tem-se o conjunto dos tipos de veículos $M = \{1, \dots, m\}$. Para cada tipo $k \in M$, m_k veículos são disponíveis no depósito, ou melhor, m_k é a quantidade de veículos do tipo k. Além disso, a capacidade de cada tipo de veículo k é denotada por Q_k e um custo fixo F_k associado que representa o aluguel ou custo de amortização. Quando um veículo sai para entrega, existirá um fluxo de variável (variável de decisão no modelo) y_{ij} especificando a quantidade de mercadorias que este veículo carrega quando vai de um cliente i para um cliente j. E também, para cada arco $(i, j) \in A$ e para cada tipo $k \in M$, considera-se um custo de roteamento não negativo c_{ij}^k. Além do mais, existem também as variáveis de decisão x_{ij}^k que assumem valores binários de 0 ou 1. Dado um veículo $k \in M$ e um arco $(i, j) \in A$, se nas rotas de solução, este veículo deverá passar por este arco, então o valor da variável é 1, caso contrário, será nula.

2.2.2 Formulação Matemática

As formulações matemáticas para HVRPFD são diversas, porém o que será apresentado a seguir é a variante mais geral em [4].

$$(1) \quad Min \sum_{k \in M} F_k \sum_{j \in V'} x_{0j}^k + \sum_{k \in M} \sum_{\substack{i,j \in V \\ i \neq j}} c_{ij}^k x_{ij}^k$$

De tal modo que

$$(2) \sum_{k \in M} \sum_{i \in V} x_{ij}^k = 1, \forall j \in V'$$

$$(3) \sum_{i \in V} x_{ip}^k - \sum_{j \in V} x_{pj}^k = 0, \forall p \in V', \forall k \in M$$

$$(4) \sum_{j \in V'} x_{0j}^k \leq m_k, \forall k \in M$$

$$(5) \sum_{i \in V} y_{ij} - \sum_{i \in V} y_{ji} = q_j, \forall j \in V'$$

$$(6)\ q_j \sum_{k \in M} x_{ij}^k \leq y_{ij} \leq (Q_k - q_i) \sum_{k \in M} x_{ij}^k \quad para\ \forall i, j \in V, i \neq j$$

$$(7)\ y_{ij} \geq 0\ para\ \forall i, j \in V, i \neq j$$

$$(8) x_{ij}^k \in \{0,1\} para\ \forall i, j \in V, i \neq j, k \in M$$

O Função Objetivo (1) é a parte que descreve a finalidade essencial da formulação. A primeira parte representa a somatória de todos custos fixos dos veículos que serão utilizados para entrega. Já na segunda parte, é a somatória de todo o custo de roteamento do trajeto percorrido por todos os veículos que saíram do depósito. Logo, a adição das duas partes será o custo total para atender todos clientes. E de uma gama de possibilidades e de soluções, busca-se o custo que seja o mínimo possível.

As restrições (2) e (3) garantem, respectivamente, que um único cliente é visitado uma única vez e que ao visitar um cliente, com certeza irá para outro local, além disso, o retorno do veículo ao depósito também é assegurado.

As restrições (4) garantem que não exceda o número que se tem de veículos de um mesmo tipo

As restrições (5) representam o balanceamento de carga de cada veículo após a visita de um cliente, mais especificamente, é a igualdade da diferença de mercadoria antes e após a visita de um cliente com a demanda desse cliente.

As restrições (6) garante que a capacidade do veículo não seja ultrapassada. A desigualdade diz que a demanda do próximo cliente tem que ser menor ou igual do que a quantidade que o veículo tem no momento, o qual essa quantidade é menor ou igual a capacidade do veículo subtraído pela entrega anterior.

As restrições (7) determinam que toda variação de fluxo y_{ij} tem de ser positivo, pois, caso contrário, um veículo carregaria uma quantidade negativa de mercadorias, a qual não seria coerente.

As restrições (8) especificam a variável de decisão x_{ij}^k como binária e restringe a possibilidade de arcos com nós iguais ($i = j$) semelhantes aos dois itens anteriores, ou seja, não é possível trajetos de um nó a ele mesmo.

2.2.3 Implementação na linguagem Python

O modelo descrito na Seção 2.2.2 foi computacionalmente implementado utilizando a linguagem de programação Python. O Python escolhido é da versão 3.8.10, utiliza-se um ambiente de desenvolvimento integrado de plataforma cruzada de código aberto denominado Spyder para realizar a implementação. É importante também a inclusão de algumas bibliotecas, essas bibliotecas contêm módulos embutidos (escritos em C) que fornecem acesso à funcionalidade do sistema, e podem ser facilmente baixados e integrados usando o comando script "*pip install + nome da biblioteca*". Para o modelo, foi necessário a biblioteca Pandas que serve para manipulação e análise de dados, essencial para gerenciamento de arquivos, e também, a biblioteca DOcplex que serve para resolver problemas de otimizações matemáticas. O DOcplex é fornecido pelo CPLEX Optimization Studio, por conseguinte, a instalação deste é indispensável, e no caso, foi

optada a versão 20.1.0. Mais detalhes sobre a linguagem Python podem ser encontrados em [5] e [6].

As primeiras partes da implementação têm foco em preparar toda informação necessária para solucionar o problema. Inicialmente, observa-se as importações das bibliotecas, criação do modelo e gerenciamento dos arquivos, e no caso, os dados estavam em Excel.

```
1  #Dados e variaveis
2  import pandas as pd
3  from docplex.mp.model import Model
4  md1 = Model('Problema de roteamento de Veículo')
5  excel = pd.ExcelFile(r'C:\Users\User\Desktop\Dados HVRPFD.xlsx')
6  demanda = pd.read_excel(excel,'Demanda')
7  rotas = pd.read_excel(excel,'CustosRotas')
8  veiculos = pd.read_excel(excel,'CustosVeiculos')
9
10 #CONJUNTOS
11 V = [n for n in range(len(demanda))]
12 A = [(i,j) for i in range(len(V)) for j in range(len(V)) if i!=j]
13 V2 = [n for n in range(1,len(demanda))]
14 D = {n: row['Demanda'] for n, row in demanda.iterrows()}
15 M = [m for m in range(len(veiculos))]
16 M_K = {n: row['Quantidade'] for n, row in veiculos.iterrows()}
17 Q_K = {n: row['Capacidade'] for n, row in veiculos.iterrows()}
18 F_K = {n: row['Custos'] for n, row in veiculos.iterrows()}
19 arcos = [(i,j,k) for k in M for (i,j) in A]
20 C_rascunho = {n: row['Custos'] for n, row in rotas.iterrows()}
21 C = {arcos[k]:C_rascunho[k] for k in C_rascunho}
22 X = {(i,j,k): md1.binary_var(name="x_{0}_{1}_{2}".format(i,j,k)) for k in M for (i,j) in A}
```

Figura 1: Implementação em Python linha 1 a 22

E nas últimas partes, são as especificações da função objetivo, restrições e a solução do problema.

```
23
24 #Funcao Objetivo
25 s_1 = md1.sum(F_K[k]*X[(0,j,k)] for k in M for j in V2)          #PRIMEIRA PARTE DA FORMULAÇÃO
26 s_2 = md1.sum([(i,j,k)]*X[(i,j,k)] for (i,j,k) in arcos)         #SEGUNDA PARTE DA FORMULAÇÃO
27 md1.minimize(s_1+s_2)
28
29 #RESTRICOES
30 #Restricao(2) TODO CLIENTE É VISITADO UMA ÚNICA VEZ
31 for j in V2:
32     md1.add_constraint(md1.sum(X[(i,j,k)] for k in M for i in V if i!=j ) == 1)
33
34 #Restricao(3) GARANTIR A SAÍDA AO EFETUAR A ENTRADA EM UM NÓ
35 for k in M:
36     for p in V2:
37         md1.add_constraint(md1.sum(X[(i,p,k)] for i in V if i!=p ) == md1.sum(X[(p,j,k)] for j in V if j!=p))
38
39 #Restricao(4) NAO EXCEDER QUANTIDADE DE VEICULO DO MESMO TIPO
40 for k in M:
41     md1.add_constraint(md1.sum(X[(0,j,k)] for j in V2)<=M_K[k])
42
43 #Restricao(5) DIFERENCA DA MERCADORIA ANTES E APOS A VISITA DE UM CLIENTE
44 Y = {(i,j): md1.integer_var(name="y_{0}_{1}".format(i,j)) for (i,j) in A}
45 for j in V2:
46     md1.add_constraint(md1.sum(Y[(i,j)] for i in V if i!=j) - md1.sum(Y[(j,i)] for i in V if j!=i) == D[j])
47
```

Figura 2: Implementação em Python linha 23 a 47

```
48 #Restricao(6) Garante que a capacidade do veiculo não seja ultrapassada
49 for (i,j) in A:
50     md1.add_constraint(D[j]*md1.sum(X[(i,j,k)] for k in M)<= Y[(i,j)] )
51     md1.add_constraint(Y[(i,j)] <= (Q_K[k]-D[i])*md1.sum(X[(i,j,k)] for k in M))
52
53 #Restricao(7) O fluxo de mercadoria sempre positivo
54 for (i,j) in A:
55     md1.add_constraint(Y[(i,j)]>=0)
56
57 #Restricao(8) serve para garantir que Xijk seja binario,
58 # mas ja foi definido assim, logo nao será necessario essa restricao
59
60 print(md1.export_to_string())
61
62 #RESOLVENDO O PROBLEMA
63 solucao=md1.solve(log_output=True)
64 print(md1.get_solve_status())
65 solucao.display()
```

Figura 3: Implementação em Python linha 48 a 65

2.3 A construção de uma heurística construtiva gulosa

Em definição computacional, heurística é um conjunto de regras e métodos que conduzem à descoberta, à invenção e à resolução de problemas. As heurísticas propõem procedimentos mentais que auxiliam a encontrar resposta adequadas ao problema, por mais que nem sempre são as ótimas, mas geralmente reduz o tempo na busca de soluções. No problema de HVRPFD, resolvendo pela linguagem Python usando DOcplex, o programa provavelmente irá contabilizar todos os custos possíveis e depois escolher o mínimo dentre os encontrados. No entanto, para instâncias grandes que geram um número gigantesco de possibilidades, a eficiência do programa será reduzida. Nesta perspectiva, as heurísticas costumam estabelecer critérios que diminuam essa quantidade de escolhas, e que ao mesmo tempo, atenda a função objetivo.

A heurística construtiva gulosa será o foco do trabalho. O objetivo é a cada passo procurar estender a solução pelo menor custo. Em outras palavras, dado um veículo $\boldsymbol{k}$ num nó $\boldsymbol{i}$, deverá escolher seu próximo destino, nó $\boldsymbol{j}$, de forma que o arco formado $(\boldsymbol{i}, \boldsymbol{j})$ tenha o menor custo de roteamento $\boldsymbol{c_{ij}^{k}}$ entre todas as rotas possíveis.

A implementação da heurística construtiva gulosa também foi feita pela linguagem Python, apresentando semelhanças na primeira parte de fornecer as informações necessárias, mas diverge pela dispensabilidade do uso do DOcplex.

```
1  #Dados e variaveis
2  import pandas as pd
3  excel = pd.ExcelFile(r'C:\Users\User\Desktop\Dados HVRPFD.xlsx')
4  demanda = pd.read_excel(excel,'Demanda')
5  rotas = pd.read_excel(excel,'CustosRotas')
6  veiculos = pd.read_excel(excel,'CustosVeiculos')
7
8  #CONJUNTOS
9  V = [n for n in range(len(demanda))]
10 A = [(i,j) for i in range(len(V)) for j in range(len(V)) if i!=j]
11 V2 = [n for n in range(1,len(demanda))]
12 D = {n: row['Demanda'] for n, row in demanda.iterrows()}
13 M = [m for m in range(len(veiculos))]
14 M_K = {n: row['Quantidade'] for n, row in veiculos.iterrows()}
15 Q_K = {n: row['Capacidade'] for n, row in veiculos.iterrows()}
16 F_K = {n: row['Custos'] for n, row in veiculos.iterrows()}
17 arcos = [(i,j,k) for k in M for (i,j) in A]
18 C_rascunho = {n: row['Custos'] for n, row in rotas.iterrows()}
19 C = {arcos[k]:C_rascunho[k] for k in C_rascunho}
```

Figura 4: Heurística em Python linha 1 a 19

Em seguida, foram definidas 4 funções no Python para a Heurística:

(i) A definição do mínimo de um dicionário (conjunto de dados associado a um valor), utilizado para o programa retornar o menor valor dos dados com os custos de roteamento.

```
20
21 def minimo(dicionario):
22     minval = min(dicionario.values())
23     res = [k for k, v in dicionario.items() if v==minval]
24     return res
25
```

Figura 5: Heurística em Python linha 20 a 25

(ii) A definição de uma heurística gulosa (HG) que fixa um veículo e procura a sua rota viável de maneira que todo próximo cliente visitado tenha o menor custo de roteamento, além de verificar sempre a sua capacidade disponível em relação a demanda do cliente. E por fim, retorna a rota inteira que o veículo irá percorrer na forma de uma lista de arcos.

```
26 def HG(veiculo_k):
27     x=Q_K[veiculo_k]
28     Cliente=V2
29     ROTA=[]
30     Cliente_P=[]
31     n=0
32     for j in Cliente:
33         if D[j]<=x:
34             Cliente_P.append(j)
35     while len(Cliente_P)!=0:
36         CA={(n,j,veiculo_k):C[(n,j,veiculo_k)] for j in Cliente_P}
37         m = minimo(CA)
38         ROTA.append(m[0])
39         for (i,j,k) in CA:
40             if (i,j,k)==m[0]:
41                 n=j
42                 x=x-D[j]
43                 Cliente_P.remove(j)
44         for j in Cliente_P:
45             if D[j]>x:
46                 Cliente_P.remove(j)
47     (a,b,c)=ROTA[-1]
48     ROTA.append((b,0,veiculo_k))
49     return ROTA
```

Figura 6: Heurística em Python linha 26 a 49

(iii) A definição que retorna todo custo de roteamento percorrida por um veículo.

```
50
51  def Custo_Rota(veiculo_k):
52      R=HG(veiculo_k)
53      Custo_Rota=0
54      for (i,j,k) in R:
55          Custo_Rota= Custo_Rota+C[(i,j,k)]
56      return Custo_Rota
57
```

Figura 7: Heurística em Python linha 50 a 57

(iv) A definição que retorna os clientes atendidos por um veículo

```
58  def Clientes_da_Rota(veiculo_k):
59      R=HG(veiculo_k)
60      CdR=[]
61      for (i,j,k) in R:
62          if (i,j,k)!=R[-1]:
63              CdR.append(j)
64      return CdR
65
```

Figura 8: Heurística em Python linha 58 a 65

No final, com a lista de todos os veículos disponíveis no estacionamento (organizada de ordem decrescente em relação a capacidade do veículo), define-se o método de solução. Enquanto ainda tiver clientes que não foram atendidas, deve-se mandar veículos na ordem da lista construída anteriormente. E por fim, retornar o custo total que seria a soma dos custos fixos de todos veículos usados e os custos de roteamento dos trajetos.

```
66  TdV=[]
67  CT=0
68  for i in M_K:
69      for u in range(M_K[i]):
70          TdV.append(i)
71  while len(V2)!=0:
72      print(HG(TdV[0]))
73      CT = CT + F_K[TdV[0]] + Custo_Rota(TdV[0])
74      CdR = Clientes_da_Rota(TdV[0])
75      for y in CdR:
76          V2.remove(y)
77      TdV.pop(0)
```

Figura 9: Heurística em Python linha 66 a 77

2.4 Resultados computacionais preliminares

Para fazer um primeiro estudo sobre o modelo e a heurística criada em conjunto, foi feito testes com 10 instâncias, com dados criados aleatoriamente, para comparar os resultados obtidos. Nos dados das instâncias que serão apresentadas, terá as informações sobre:

- **O número de clientes (n)**
- **A quantidade de veículos de tipos diferentes (V_n)**
- **Custo fixo de cada tipo de veículo**
- **Quantidade de cada tipo de veículo**
- **Capacidade de cada tipo de veículo**
- **Demanda dos clientes**
- **Custo de Roteamento**

Na parte do custo de roteamento, é uma lista que respeita os valores dos nós de partida na ordem crescente, descreve todos os custos para um veículo e depois para um outro. Por exemplo, considere primeiramente que (i, j, k) representa a rota de i a j com o veículo k, suponha que temos 2 clientes e 2 tipos de veículos, o arco correspondente à lista dos custos de roteamento seria:
(0,1,1) (0,2,1) (1,0,1) (1,2,1) (2,0,1) (2,1,1) (0,1,2) (0,2,2) (1,0,2) (1,2,2) (2,0,2) (2,1,2)

É interessante observar que a quantidade de valores dos custos de roteamento é sempre dada pela seguinte fórmula:

$$[(n+1)^2 - (n+1)] \times V_n$$

A origem da fórmula se deve ao fato de que tem $(n+1)$ nós (contando o depósito), e então, a quantidade de arcos possíveis seria $(n+1)^2$, porém os arcos com nós iguais são inválidos, logo subtrai-se $(n+1)$ nós. Até aqui, teria a quantidade de rotas totais, porém segundo a terceira característica comentada na Seção 2.1, considera-se diferentes custos para veículos distintos, logo explica-se a presença da multiplicação. Vale salientar que arcos com sentidos opostos (i, j) e (j, i) tem o mesmo custo de roteamento dado um veículo k.

Segue os dados de cada instância:

- **Instância 1**

5 Clientes
2 Tipos de veículos
Custo fixo de cada tipo de veículo 65 85
Quantidade de cada tipo de veículo 02 01
Capacidade de cada tipo de veículo 10 12
Demandas dos clientes 04 05 04 07 06
Custo de Roteamento (60 valores)
03 04 03 02 04 03 05 03 06 04 04 05 03 04 04 03 03 03 05 02 02 06 04 05 03 04 04 04 02 03 02 04 03 02 03 02 03 05 04 05 04 03 02 03 02 03 05 02 04 03 02 04 03 04 04 03 05 02 03 04

- **Instância 2**

5 Clientes
3 Tipos de veículos
Custo fixo de cada tipo de veículo 65 70 85
Quantidade de cada tipo de veículo 01 01 01
Capacidade de cada tipo de veículo 10 11 12
Demandas dos clientes 04 05 08 03 04
Custo de Roteamento (90 valores)
05 03 09 07 02 05 08 01 04 06 03 08 05 06 02 09 01 05 03 08 07 04 06 03 07 02 06 02 08 07 04 04 07 09 06 04 03 01 08 02 04 03 05 09 07 07 01 05 05 06 09 08 09 05 02 06 02 07 06 02 08 04 01 03 09 08 03 02 05 07 04 03 06 01 04 01 02 06 08 09 03 05 01 08 08 09 07 04 09 01

- **Instância 3**

8 Clientes
2 Tipos de veículos
Custo fixo de cada tipo de veículo 10 15
Quantidade de cada tipo de veículo 02 02
Capacidade de cada tipo de veículo 15 18
Demandas dos clientes 07 04 06 02 03 05 03 08
Custo de Roteamento (144 valores)
05 04 07 01 06 09 02 01 05 06 04 05 09 03 08 02 04 06 07 08 05 09 06 02 07 04 07 07 04 03 03 08 01 05 08 07 09 05 06 04 06 09 05 04 09 02 01 07 09 03 09 03 05 02 03 04 02 08 06 03 06 01 03 08 01 02 02 08 04 07 04 08 06 01 07 09 02 05 08 01 06 09 06 03 07 04 05 02 01 09 06 03 09 05 01 07 07 06 06 08 04 02 01 08 09 03 03 08 06 02 07 09 02 07 09 04 06 03 05 04 05 04 05 02 02 03 03 09 08 05 01 01 07 05 03 08 01 02 07 08 09 04 09 08

- **Instância 4**

8 Clientes
3 Tipos de veículos
Custo fixo de cada tipo de veículo 10 11 12
Quantidade de cada tipo de veículo 02 01 02
Capacidade de cada tipo de veículo 15 16 17
Demandas dos clientes 04 02 06 07 07 03 08 02

Custo de Roteamento (216 valores)

06 09 01 05 04 05 03 09 06 06 02 01 04 07 08 03 09 06 05 06 09 04 08 02 01 02 05 01 07 05 07 07 05 01 06 01 08 04
02 06 04 04 09 07 08 09 03 09 05 07 04 05 04 09 05 06 03 08 08 07 02 03 05 07 09 03 02 07 06 09 06 07 03 08 01 04
02 09 03 04 03 02 06 08 07 01 05 04 08 02 08 01 03 06 02 07 01 06 08 09 05 04 06 03 04 08 01 09 05 08 02 07 02 07
03 05 05 09 01 02 09 01 06 04 08 09 04 07 03 05 02 06 02 01 04 03 04 04 07 03 07 02 07 03 06 09 01 08 05 04 09 05
06 06 02 08 01 07 03 03 09 06 06 08 05 01 02 07 01 02 06 04 09 05 03 04 08 08 08 04 09 02 07 06 05 01 05 09 09 01
08 05 04 07 01 05 02 01 06 01 09 03 02 03 07 08 06 08 05 03 07 04 06 05 01 08

- **Instância 5**

10 Clientes

2 Tipos de veículos

Custo fixo de cada tipo de veículo 12 15

Quantidade de cada tipo de veículo 02 03

Capacidade de cada tipo de veículo 25 35

Demandas dos clientes 18 14 09 12 07 15 20 06 17 16

Custo de Roteamento (220 valores)

17 12 11 18 20 15 13 16 10 19 17 13 16 12 14 11 15 20 17 10 12 13 19 18 12 20 13 11 18 15 11 16 19 16 12 19 17 10
17 12 18 12 18 16 18 10 11 15 14 19 20 14 12 12 18 16 20 11 17 14 15 11 20 19 10 16 16 12 18 11 13 15 13 17 11 20
16 15 19 13 16 20 11 10 15 11 12 15 18 20 10 17 18 17 14 17 18 19 18 14 19 10 15 12 19 14 11 13 20 14 10 16 13 15
17 12 19 11 13 20 10 15 12 11 19 14 16 17 18 10 16 15 14 16 19 20 11 18 13 17 13 12 14 10 12 14 15 10 19 13 15 11
16 10 17 12 18 20 16 15 17 19 19 12 17 14 11 17 15 18 12 14 20 14 12 14 10 20 12 11 19 16 11 15 18 11 10 16 15 13
11 17 18 10 20 17 20 16 19 14 13 18 13 19 16 15 12 15 19 13 20 10 17 13 15 18 11 13 14 13

- **Instância 6**

10 Clientes

3 Tipos de veículos

Custo fixo de cada tipo de veículo 14 16 18

Quantidade de cada tipo de veículo 02 02 01

Capacidade de cada tipo de veículo 23 26 29

Demandas dos clientes 13 20 06 05 09 19 14 16 07 18

Custo de Roteamento (330 valores)

22 24 21 18 10 13 12 15 11 23 22 17 19 20 16 14 25 11 24 20 24 17 19 15 16 17 23 25 14 21 21 20 19 22 18 12 13 10
18 10 18 19 15 22 24 22 15 16 11 17 10 16 16 18 24 23 25 13 21 14 13 14 17 12 22 23 12 19 11 19 12 25 23 13 15 25
12 15 10 22 15 11 25 10 16 13 19 15 21 20 11 24 14 18 11 21 11 10 21 23 23 20 21 10 17 14 19 22 20 23 18 16 15 14
24 13 12 25 17 25 18 24 16 21 19 18 20 15 17 10 16 24 13 11 12 14 23 20 22 21 15 16 13 16 17 19 12 10 25 23 14 21
11 16 15 18 24 14 11 13 24 19 12 17 15 11 18 10 22 14 13 18 14 19 18 11 12 23 19 14 12 20 23 12 24 18 12 24 20 25
25 15 20 10 14 10 23 24 13 15 17 17 22 25 11 22 19 20 13 21 25 10 21 23 13 14 14 25 15 21 17 18 20 22 19 12 11 17
16 10 17 23 13 25 24 14 15 21 11 17 18 23 23 15 19 22 25 13 21 14 20 13 23 16 20 24 12 10 22 23 22 25 15 16 20 11
14 18 15 10 19 24 19 20 20 13 25 17 19 12 12 14 22 24 21 13 16 24 23 21 11 15 25 12 14 25 16 24 18 15 17 21 23 10
18 17 24 24 11 22 16 11 21 22 15 19 23 18 11 25 10 17 14 23 14 12 21 15 22 25

- **Instância 7**

12 Clientes

2 Tipos de veículos

Custo fixo de cada tipo de veículo 25 40

Quantidade de cada tipo de veículo 04 02

Capacidade de cada tipo de veículo 20 40

Demandas dos clientes 18 02 09 19 11 08 16 14 04 05 06 13

Custo de Roteamento (312 valores)

13 15 20 18 11 14 10 16 15 12 22 23 13 21 19 24 17 21 15 23 25 16 20 19 15 21 18 22 24 13 11 10 12 17 14 11 20 19
18 16 15 20 10 12 13 22 23 21 18 24 22 16 14 18 17 24 25 22 21 17 11 17 24 15 14 20 15 12 19 13 14 23 14 21 13 20
18 20 25 10 24 16 11 18 10 15 11 10 17 15 25 15 22 17 14 18 16 23 10 12 24 12 10 15 11 25 20 23 15 25 12 13 25 19
24 22 11 19 20 16 12 16 17 22 22 13 16 17 25 19 11 24 22 20 14 23 21 14 11 14 20 20 11 12 23 19 11 21 17 23 18 18
23 16 24 12 25 20 16 18 24 11 23 22 13 12 19 15 25 17 21 24 23 18 11 20 13 16 19 12 20 17 14 21 25 17 15 22 24 10
20 10 16 21 14 24 19 13 23 16 22 21 17 11 18 24 21 24 15 18 12 25 14 17 18 14 24 23 25 19 15 24 25 14 12 20 19 13
11 28 17 13 18 24 21 16 10 25 14 12 23 11 15 23 12 25 21 23 19 25 22 24 22 20 22 16 25 14 16 23 16 17 20 10 13 13
24 22 14 12 10 19 16 21 15 13 12 16 10 21 17 20 25 25 17 21 11 18 19 19 20 17 18 19 14 22 20 15 11 14 15 12 10 11
14 13 12 24 10 13 18 14

- **Instância 8**

12 Clientes

3 Tipos de veículos

Custo fixo de cada tipo de veículo 15 18 40

Quantidade de cada tipo de veículo 02 05 03

Capacidade de cada tipo de veículo 50 60 80

Demandas dos clientes 30 23 15 14 03 05 21 06 04 22 08 19

Custo de Roteamento (468 valores)

22 26 21 17 11 24 19 16 30 15 28 18 22 12 13 20 23 14 15 25 27 29 26 30 26 12 21 19 22 17 11 29 12 28 23 14 21 13
21 13 18 16 25 15 24 20 14 24 17 20 19 13 29 12 19 22 16 30 11 18 11 23 22 18 29 23 28 25 15 27 17 20 24 14 17 16
12 23 13 26 20 25 30 14 19 15 11 25 19 28 13 24 21 23 13 11 16 25 29 15 22 25 26 24 16 17 28 26 30 27 12 24 16 15
20 21 16 29 12 22 15 29 28 20 30 27 25 23 17 29 15 18 28 26 23 14 11 17 30 13 28 12 15 27 18 30 14 24 18 20 14 11
26 22 18 27 19 30 23 16 22 21 12 29 14 25 11 27 19 24 18 19 26 17 28 15 20 13 16 15 30 24 11 24 19 13 28 20 27 18
24 20 23 18 11 23 25 17 22 12 26 21 15 21 16 19 24 23 29 23 30 26 17 18 22 25 22 26 19 25 29 13 24 28 27 16 14 19
21 17 13 17 23 13 11 12 20 23 16 20 12 28 28 22 30 24 11 18 26 28 14 17 29 15 20 12 26 28 12 18 19 21 24 12 14 20
27 26 17 27 20 26 19 27 15 30 25 13 18 21 18 16 23 28 21 27 29 22 11 16 24 15 22 14 16 14 14 15 29 25 27 15 20 21
25 19 20 17 12 30 22 25 11 29 18 25 15 23 16 28 12 19 20 13 11 17 26 30 14 27 22 24 11 21 20 17 29 17 14 29 23 15
12 30 21 27 11 19 18 26 14 16 26 22 18 13 24 28 25 24 25 30 29 16 18 14 11 23 21 20 26 19 15 14 23 26 18 13 12 15
30 29 25 25 23 27 15 22 14 13 28 17 22 27 16 11 16 22 12 18 11 12 28 25 22 28 12 18 28 24 30 13 23 15 17 25 19 21
20 23 12 11 21 24 21 30 22 22 19 17 14 30 19 21 27 28 20 29 27 28 21 17 15 29 20 20 11 25 26 25 16 12 20 14 15 16
13 17 19 24 19 25 11 18 23 30 29 16

- **Instância 9**

15 Clientes

2 Tipos de veículos

Custo fixo de cada tipo de veículo 30 50

Quantidade de cada tipo de veículo 07 04

Capacidade de cada tipo de veículo 40 60

Demandas dos clientes

22 11 36 26 31 35 07 33 08 25 10 32 12 18 23

Custo de Roteamento (480 valores)

37 16 32 29 13 20 36 22 14 21 23 29 10 11 17 37 26 38 15 28 30 27 35 40 19 31 12 33 24 34 16 26 18 25 16 10 13 14
26 37 32 31 39 30 38 32 38 18 18 29 36 23 19 35 11 17 12 33 15 40 29 15 25 18 28 34 25 27 22 21 24 20 18 16 38 13
28 16 29 28 13 19 35 31 21 23 38 22 40 17 20 30 10 36 34 13 25 34 28 12 20 33 14 36 10 36 27 13 23 25 19 25 24 32
39 26 27 11 30 15 22 35 14 19 27 35 34 24 28 35 26 27 33 39 13 14 40 26 35 22 31 28 32 28 17 31 19 40 34 20 21 19
37 11 21 21 12 39 35 17 10 12 24 15 38 23 31 32 17 24 23 20 26 26 31 10 18 30 16 12 29 12 31 12 20 38 33 27 27 19
12 18 11 14 21 10 33 39 33 18 22 14 11 33 40 24 30 11 23 36 11 24 30 15 16 40 36 30 39 34 15 16 14 23 32 17 34 38
40 38 17 10 15 13 20 38 12 21 36 32 25 37 33 22 11 15 23 12 31 26 34 27 28 30 32 25 19 37 11 38 20 40 29 10 18 16
24 13 21 14 37 19 39 25 35 36 40 30 24 10 37 15 16 23 11 33 37 39 20 19 25 35 32 14 21 31 22 36 18 12 22 11 25 20
34 26 39 38 13 29 27 33 28 17 18 11 38 35 19 34 11 23 27 24 35 12 14 28 36 39 15 20 36 25 26 11 15 29 34 17 22 21
38 32 31 23 40 40 35 39 23 15 33 10 13 20 37 19 30 26 12 29 30 32 38 27 29 33 16 23 31 12 17 32 15 31 10 24 14 13
24 34 10 16 22 35 28 16 38 19 26 18 10 21 29 35 17 13 23 22 26 14 34 40 36 34 16 37 31 27 12 22 20 31 35 26 37 29
21 33 27 24 15 22 33 14 21 37 12 28 14 37 24 39 30 28 13 16 36 28 28 38 19 17 16 34 29 24 10 27 30 21 23 18 17 36
32 30 32 38 40 21 39 10 18 32 14 11 12 18 39 31 26 15 19 36 33 30 27 18

- **Instância 10**

15 Clientes

3 Tipos de veículos

Custo fixo de cada tipo de veículo 30 40 50

Quantidade de cada tipo de veículo 05 02 03

Capacidade de cada tipo de veículo 70 80 90

Demandas dos clientes

30 37 15 34 19 17 22 31 39 27 40 18 36 24 20

Custo de Roteamento (720 valores)

37 39 31 26 42 13 20 30 40 50 27 35 17 19 49 37 45 21 33 48 33 22 43 47 44 24 41 10 25 15 39 45 46 12 18 38 34 29
11 16 32 36 28 21 33 31 21 46 42 43 45 31 34 17 27 14 49 26 22 29 26 33 12 42 19 11 39 48 13 38 28 32 46 15 12 42
48 18 43 19 10 50 37 18 20 40 44 35 47 36 13 33 38 45 11 10 23 16 30 24 25 41 19 32 37 20 22 34 31 39 50 23 41 46
47 44 17 39 45 35 30 43 29 34 48 37 16 41 34 18 30 22 33 20 10 40 47 11 17 13 18 30 46 34 23 27 21 11 49 13 50 44
16 27 38 20 24 47 18 23 40 31 24 25 28 27 24 32 14 28 40 25 44 30 27 40 36 43 16 14 35 41 36 49 32 44 41 17 22 21
31 36 48 50 12 17 10 28 26 46 35 19 39 33 11 24 43 48 26 15 19 25 21 22 15 47 32 45 20 49 25 16 50 26 29 49 15 33
29 12 36 37 35 10 13 28 14 12 15 29 42 38 46 27 15 33 35 40 30 28 12 11 10 39 13 42 26 18 34 37 42 45 23 49 14 24
29 20 36 25 38 26 32 16 22 50 21 19 47 41 38 43 48 17 31 46 18 32 44 23 50 47 17 15 28 38 39 35 22 42 27 34 16 44
30 13 26 32 37 43 24 29 49 44 20 15 37 22 23 30 25 12 16 36 46 41 19 18 33 10 33 42 50 50 13 25 10 27 34 45 11 21
48 31 40 35 45 21 47 26 12 10 14 48 28 26 47 45 31 44 40 23 19 17 32 16 27 14 13 10 35 11 22 20 36 30 49 47 15 37
36 34 48 13 23 32 27 43 14 19 28 14 41 28 43 46 45 28 10 23 21 12 18 42 17 12 24 38 38 24 41 11 26 35 32 21 34 33
29 50 11 29 43 39 29 19 21 47 11 27 12 34 40 24 15 10 20 48 35 49 18 48 45 22 43 18 33 40 41 16 39 36 17 22 44 33
31 31 20 14 42 29 24 41 49 13 25 31 42 20 10 40 44 36 19 17 50 15 16 49 37 25 30 38 39 46 26 19 50 41 44 27 17 25
13 37 32 12 21 39 22 48 36 10 14 47 49 15 28 11 25 32 35 46 18 34 20 31 45 42 33 38 29 24 16 30 12 35 40 43 37 23
30 45 23 34 27 16 36 42 38 21 46 40 43 50 12 26 10 12 21 44 31 35 37 39 39 18 43 43 20 15 40 14 33 48 49 47 32 24
46 22 34 37 50 20 18 22 29 38 17 11 25 19 39 26 48 20 23 12 15 18 18 41 46 12 47 28 18 16 19 36 31 30 26 40 22 18
14 39 31 14 36 17 20 50 10 45 45 10 14 29 41 14 19 21 20 17 15 42 41 14 42 23 12 33 38 46 39 19 30 48 43 24 11 44
47 33 34 21 48 17 12 31 21 30 49 37 13 23 27 49 38 27 44 49 11 47 14 20 48 49 22 33 45 17 15 29 16 31 47 25 28 36
17 43 37 22 40 10 25 28 24 36 35 32 19 18 17 15 24 13 33 40 50 13 11 16 42 37 24 39 16 20 42 11 23 45 10 50

Resultados obtidos após aplicar as instâncias no modelo matemático e na heurística construtiva gulosa:

Instância	Número de Clientes	Tipos de Veículo	Solução pelo Modelo	Limitante dual	Tempo (Modelo)	Solução pela Heurística	Tempo (Heurística)
1	5	2	237	237	0.11s	241	0.03s
2	5	3	164	164	0.14s	247	0,06s
3	8	2	60	56,53	1.31s	86	0.07s
4	8	3	59	51,62	1.72s	82	0.06s
5	10	2	233	225,05	1.77s	249	0.06s
6	10	3	282	265,49	2.23s	292	0.09s
7	12	2	303	296,96	5.48s	394	0.1s
8	12	3	250	234,29	15.89s	283	0.07s
9	15	2	548	536,98	13.96s	606	0.09s
10	15	3	521	514,67	24.28s	600	0.16s

Figura 10: Tabela com os resultados computacionais preliminares

3. Resultados parciais/preliminares obtidos

Em primeiro lugar, é o sucesso na implementação do modelo matemático na linguagem Python. No processo de implementação, houve a adaptação nas restrições do modelo da literatura, uma vez que verificou uma inconsistência na restrição (6) apresentada na Seção 2.2.2.

Em segunda lugar, é o sucesso na criação da heurística construtiva gulosa. Não foi encontrado heurísticas específicas para HVRPFD nas literaturas, e então foi possível uma criação com mais originalidade.

E por último, são os resultados computacionais preliminares apresentados anteriormente. Analisando as soluções encontradas pelos dois métodos, pode-se afirmar que a heurística construída apresenta uma eficiência razoável. Pode-se até considerar que com a ausência de procedimentos heurísticos de refinamento, os resultados são aceitáveis, além de obter sempre uma resposta mais rápida do que o modelo matemático. Mas as diferenças de valores obtidas não são desprezáveis, na visão de que foi trabalhada instâncias pequenas, e assim, não seria muito equivocado preocupar com dados maiores.

4. Trabalhos em andamento e futuros

A formulação matemática e a heurística construtiva gulosa já estão implementadas, porém para a heurística construída, as suas soluções nem sempre são as ótimas, uma vez que foi dispensado os aspectos dos custos fixos dos veículos e a soma total do custo de roteamento, como pode-se ver na figura 10. Dessa forma, a próxima etapa seria inserir procedimentos heurísticos de refinamento, ajustando a solução factível encontrada para soluções melhores, e até ótimas. E posteriormente, fazer testes com instâncias de tamanhos variados para comparar novamente os resultados tanto na qualidade da solução quanto na quantidade de tempo gasto.

As instâncias apresentadas na Seção 2.4, não pertencem a literatura, e assim, desejando testes mais significativos, serão usadas as instâncias do Loggi BUD para testar os métodos heurísticos.

Dependendo dos resultados heurísticos após os procedimentos de melhoria, pode ser que necessite a alteração da heurística construtiva gulosa. No processo dos testes das instâncias anteriores, houve ocasiões em que a heurística não retornava respostas, isso provavelmente ocorreu devido a falhas na construção do algoritmo. Segundo a lógica de construção exposta na Seção 2.3, a capacidade dos veículos não será aproveitada ao máximo. Por exemplo, temos dois veículos com capacidade 10 e 5, e clientes com demanda 3,5,4 e 3. É fácil observar que a distribuição ideal seria para o veículo de capacidade 10, 3+3+4, e para outro veículo, 5. Porém, suponha que segundo o critério guloso da heurística, o veículo de capacidade 10 optou pelos clientes com demanda 3, e depois, 5, e assim, já totaliza 8 e restando 2, e neste momento, observe que não terá como visitar mais nenhum outro cliente, apenas resta-lhe a opção de voltar ao depósito. Por outro lado, o outro veículo não conseguirá terminar as entregas, e consequentemente, a heurística não conseguirá achar solução. Portanto, conforme os futuros desenvolvimentos do trabalho, a retificação dessas falhas poderá ser possível.

AGRADECIMENTOS

Primeiramente, agradeço a empresa Loggi por contribuir por essa oportunidade de aprendizagem, são estudos inesperados na minha formação acadêmica, mas que indubitavelmente permitiu que eu adquirisse novas habilidades e conhecimentos. Em segundo lugar, agradeço imensamente ao orientador desta iniciação científica, que sempre com tamanha paciência, sanava minhas dúvidas e me acalmava quando sentia desespero em trabalhar com temas que não eram familiares. Por fim, vale citar a contribuição de todos professores do meu curso que me propiciaram base suficiente para trabalhar com esses problemas de otimizações, principalmente, a professora que ensinou a linguagem Python.

REFERÊNCIA

[1] Silvio Alexandre de Araujo; Socorro Rangel. Matemática Aplicada ao Planejamento da Produção e Logística. São Carlos - SP, Brasil. Volume 76, 2014.

[2] Professor Volmir Eugênio Wilhelm. Problema do Caixeiro Viajante. Departamento de Engenharia de Produção – UFPR. Disponível no link: < https://docs.ufpr.br/~volmir/PO_II_12_TSP.pdf>

[3] Corinne Brucato. The Traveling Salesman Problem. University of Pittsburgh, 2013. Disponível no link: < https://www.mathematics.pitt.edu/sites/default/files/TSP.pdf>

[4] Bruce Golden, S.Raghavan, & E. Wasil (Eds.), The Vehicle Routing Problem: Latest Advances and New Challenges. New York: Springer Science+Business Media. DOI:10.1007/978-0-387-77778-8

[5] Caelum, ensino e inovação. Python e orientação a objetos, curso py-14. Disponível no link: < https://www.caelum.com.br/apostila/apostila-python-orientacao-a-objetos.pdf>

[6] Marcelo Aguiar; Geraldo Mauri, Rodrigo Freitas Silva. Introdução aos Métodos heurísticos de Otimização com Python. CAUFES. Março de 2018. Disponível no link: < http://marceloaguiar.pro.br/files/Introdu%E7%E3o_MH_Python.pdf >

Otimização por Meta-Heurísticas mediado por Sistemas Multiagentes aplicados ao Problema do Roteamento de Veículos Last-Mile

Johann Pinheiro Pires*
piresjohann@furg.br
Universidade Federal do Rio Grande, Campus Carreiros
Rio Grande, Rio Grande do Sul, Brasil

Eder Mateus Nunes Gonçalves
Universidade Federal do Rio Grande, Campus Carreiros
Rio Grande, Rio Grande do Sul, Brasil
edergoncalves@furg.br

ABSTRACT

Em sua versão mais clássica, o Problema de Roteamento de Veículos (VRP) está distante das condições encontradas no mundo real, em que cada empresa de logística impõe características específicas aos seus serviços oferecidos. No entanto, todas tem a comum a demanda por escalabilidade aos métodos empregados, especialmente em regiões metropolitanas. Nesse sentido, este trabalho propõe um *framework* para a instanciação de algoritmos de coalizão multiagente e meta-heurísticas de otimização, de modo a definir rotas para os veículos baseado na dinâmica dos dados de entrega, adequando-se ao contexto de cada localidade. Nesta proposta, trabalha-se com a hipótese de que o VRP pode ser modelado como um problema de coalizão entre pacotes entrega cuja função objetivo principal é a minimização do deslocamento total para entregas de encomendas. O sistema será validado com o *benchmark* Loggi BUD, que também fomentará as estratégia *data-driven* do sistema.

KEYWORDS

pesquisa operacional, roteamento de veículos, meta-heurísticas, sistemas multiagentes.

ACM Reference Format:
Johann Pinheiro Pires and Eder Mateus Nunes Gonçalves. 2021. Otimização por Meta-Heurísticas mediado por Sistemas Multiagentes aplicados ao Problema do Roteamento de Veículos Last-Mile. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUÇÃO

O problema de Roteamento de Veículos *(VRP)* é um dos problemas mais tradicionais nas áreas de analise combinatória, com objetivo de minimizar o total da distância viajada enquanto se entrega bens para clientes dispersos geograficamente, sendo que cada entregador tem como restrição o total de tempo/distância de viagem ou a capacidade do veículo [16].

De acordo com Eksioglu et al. [4], entre 1959 e 2008 foram publicados 1021 artigos em Journals tendo VRP *(Vehicle Routing Problem)* como tópico principal, e essa literatura tem crescido a uma taxa de 6% ao ano, que, em concordância dos dados disponibilizados pela Associação Brasileira de Logística (Aslog), compravam a importância econômica do tema já que a área de logística movimenta cerca de 20% do PIB nacional.

Os algoritmos de otimização estão limitados aos seus condicionantes de aplicabilidade, que diferem consideravelmente das condições encontradas no mundo real. Deste modo, problemas reais são resolvidos por meio de soluções híbridas. A consequência mais evidente é a implementação de soluções melhores do que aquelas baseadas em procedimentos de otimização simples, ou que proveem soluções sub-ótimas em menor tempo [2].

Além da dificuldade da otimização em si, problemas VRP no mundo real trabalham com grandes quantidades de dados, e atualizações desses dados. Uma das principais limitações encontradas nas soluções propostas é a escalonabilidade, um exemplo disso é encontrado em Barbucha [1] onde se é utilizado um sistema multiagentes com o objetivo de aumentar a paralelização em um ambiente distribuído [17]. Outra variante é analisada em Lv et al. [9], em que se divide as áreas em regiões fixas, buscando diminuir o custo computacional ao representar a rota como uma sequencia de regiões visitadas.

Com o intuito de aproximar as condições das soluções propostas ao contexto do mundo real VRP, as seguintes restrições devem ser consideradas [5]:

(1) Os veículos utilizados para entregas last-mile têm capacidade máxima homegênea, considerando peso, volume e quantidade de pacotes. A literatura descreve essa condição pelo termo em inglês *Capacited CRP* (CVRP);
(2) As rotas criadas não possuem o caminho de volta a origem (Open VRP);
(3) Uma vez que os pacotes não podem ser armazenados, as rotas são criadas incrementalmente a medida que novos pacotes chegam, ao centro de distribuição (VRP dinâmico);
(4) Nada se sabe sobre os pacotes a serem despachados até chegaram ao centro de distribuição. No entanto, é possível o uso de dados históricos de entregas para estratégias de previsão;
(5) Modelos devem prever a noção de janela de tempo, o intervalo de tempo em que o cliente pode ser atendido quando da entrega.

Dado este contexto, este trabalho propõe um *framework* para a instanciação de técnicas de coalizão multiagente e meta-heurísticas para otimização de rotas, de modo a escolher a melhor combinação de acordo com a região de entrega. Trabalha-se sob a hipótese de

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference'17, July 2017, Washington, DC, USA
© 2021 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

que o VRP pode ser modelado como um problema de coalizão multiagente Shehory et al. [13] entre pacotes de entrega e entregadores, tendo como função objetivo a minimização de métrica global de deslocamento para entrega. Mesmo que o sistema não conheça explicitamente o valores globais, a aplicação de coalizão distribuída multiagente permite a realização do problema.

2 TRABALHOS RELACIONADOS

Eksioglu et al. [4] em seu trabalho propôs a seguinte a taxonomia para os trabalhos de *VRP*, que foi atualizada por Braekers et al. [3]:

(1) *Tipo de Estudo:* classifica o trabalho quanto aos seus aspectos teóricos, métodos aplicados e forma de documentação;
(2) *Características de Cenários:* listando todas as características que podem ser encontradas nos cenários de problematização VRP;
(3) *Características Físicas do Problema:* são todos os fatores que estão incluídos nas restrições que diretamente afetam as soluções geradas, como a definição da rede de transporte, enquadramento dos clientes na estrutura, classificação dos pontos de origem e destino, entre outros fatores;
(4) *Características da Informação:* distingue a evolução, qualidade, disponibilidade e processamento da informação;
(5) *Características dos Dados:* quanto a origem dos dados utilizados, de do mundo-real, dados sintéticos como aqueles produzidos por benchmarks, ou uma combinação entre eles.

De acordo com Braekers et al. [3], meta-heurística tem grande participação nos trabalhos analisados, cerca de 71,25% deles. A meta-heurística mais recorrente entre as soluções para VRP é o algoritmo genético devido a sua fácil adaptação e eficiência. Um exemplo recente desta abordagem foi proposta em [11], onde um algoritmo genético com chave aleatória em bias resolve VRP com restrições de capacidade e distâncias. É empregado em Open VCR, ou seja, o veículo termina sua rota ao fazer a entrega para o último cliente, sem a necessidade de retorno ao depósito. Além disso, os veículos não podem exceder a capacidade do veículo e a distância máxima previamente estabelecida.

Outra técnica bastante utilizada é a Busca Tabu que ainda conta com grande popularidade, especialmente algumas das suas variações. Em Lai et al. [8], uma frota heterogênea de veículos é considerada como principal característica para o *VRP*, enquanto em Qiu et al. [10] a existência de entregas e buscas são consideradas como principal restrição. Em ambas abordagens, a busca tabu é implementada de maneira isolada, o que depende fortemente da implementação de estruturas de memória avançadas. Já em [14] a busca tabu é implementada junto com uma busca por vizinhança variada, de modo que o modelo considere uma distribuição de capilaridade de entregas em áreas urbanas maiores a partir de seis características: restrições de capacidade, janelas de tempo,busca e entrega, limitação de carga de trabalho, compatibilidade entre ordens, e caminhos abertos.

3 ARQUITETURA DO SISTEMA

O objetivo deste projeto é a especificação e implementação de um *framework* que permita a instanciação de diferentes estrategias de roteamento de veículos integrando métodos de otimização a mecanismos de tomada de decisão multiagentes, de modo a permitir a escolha da melhor técnica para cada região onde exista a necessidade de distribuição de entregas. Com isso, podemos enumerar os seguintes objetivos:

- Especificação e implementação de um biblioteca de agentes, onde cada agente é implementado por técnica de otimização específica, selecionada dentre aquelas mais utilizadas na literatura;
- Construção e disponibilização de middleware baseado em mecanismos de tomada de decisão multiagente para a mediação entre os agentes de otimização;
- Interface para configuração e instanciação dos agentes e dos mecanismos de mediação.

Figure 1: Sistema Multi-Agentes

Ao ser modelado como um sistema multiagente, o sistema é dinâmico e aberto, ou seja, é capaz de fazer alterações de rota devido a modificação de diferentes variáveis, como número de agentes, número de entregadores, dados de tráfego, em tempo de execução, o que inclui os critérios de otimização.

Em Galindo et al. [5] temos uma descrição do processo logístico de entregas utilizados pela empresa *Loggi*, caracterizando o contexto do problema, o que demanda uma breve explicação do processo. As entregas chegam no Centro de Distribuição , e são agrupadas em pacotes, os quais já tem rotas e entregadores vinculados, e são enviados aos centros intermediários, chamados de Centro de Expedição *(CE)*. Por ultimo, os entregadores responsáveis pegam seus respectivos pacotes e fazem as entregas seguindo a rota pré-estabelecida. Como podemos observar, todas as decisões de logística são tomadas dentro do próprio DC, sendo as outras etapas operacionais.

Com isso em mente foi desenvolvido a arquitetura do sistema multi-agentes descrita na figura 1. Re-interpretando a logica descrita nas operações, os agentes do sistema são os entregadores cadastrados, que observam o ambiente dos CD e CEs verificando os pacotes que chegam no sistema e discutindo entre si qual dentre eles que será responsável por aquela entrega. Assim montando os pacotes e definindo as rotas de entregas.

Figure 2: Sistema Multi-Agentes detalhes de implementação

A montagem dos pacotes é definido por processos de formação de coalizão entre os agentes entregadores [13]. Métodos de formação de coalizão entre múltiplos agentes são aplicados quando grupos de agentes trabalhando em conjunto cooperam para realizar suas tarefas. Deste modo, aumenta-se a eficiência do grupo na execução de tarefas, resultando em uma performance próxima a ótima.

O modelo de coalizão proposto prevê um ambiente de agentes abertos e dinâmicos. Para isso, considera um conjunto de n agentes, $\{A_1, A_2, \ldots, A_n\}$, onde cada agente A_i tem um vetor não-negativo de capacidades $B_I = \langle b_1^i, \ldots, b_r^i \rangle$. Cada capacidade é uma propriedade de um agente que quantifica sua habilidade para executar um tipo específico de ação. Existe ainda um conjunto $|T|$de tarefas independentes $T = \{t_1, t_2, \ldots, t_m\}$. Para a satisfação de cada tarefa t_j um vetor de capacidades $B_j = \langle b_1^j, \ldots, b_r^j \rangle$ é necessário. Os benefícios em termos de performance dependem das capacidades requeridas, e devem ser medidas sob a perspectiva de todo o sistema.

A coalizão é definida como um grupo de agentes que decide cooperar para executar uma tarefa em comum. A coalizão C tem um vetor de capacidades B_c que é a soma das capacidades que os membros da coalizão contribuem para uma coalizão específica. C tem um valor V que o benefício conjunto dos membros de C quando cooperam satisfatoriamente em uma tarefa específica.

A figura 2 apresenta as especificações da arquitetura, incluindo detalhes de implementação. Os agentes entregadores serão responsáveis por todo o processo de decisão e neles que será implementado os módulos para as meta-heurísticas, além dos cálculos de rotas. Também fica destacado as estruturas de dados e as restrições do sistema, sendo o ultimo: Volume; Peso; Local de entrega. As entregas chegam em forma de *Streaming* em tempo real e a saída do sistema são o conjunto de pacotes *(Unidade de Pacotes*, que será enviada para os CEs.

É importante destacar algumas decisões de projeto. Para implementação do sistema multiagente esta sendo utilizado a linguagem Python 3.8.1 em conjunto do framework SPADE *(Smart Python Agent Development Environment)*, idealizado para o desenvolvimento de sistemas multi-agentes em python baseado em mensagens instantâneas (XMPP). Na figura 3 esta representado as etapas de execução da implementação em situações normais. É importante notar que abstraímos a existência do CE, já que seu papel é nulo do ponto de vista da aplicação, sendo assim ele esta sublimado junto do DC, sendo apenas um agente.

Ao inicializar, o sistema troca mensagens com entregadores, confirmando sua existência e conferindo seus dados, que seriam as restrições de cada carro como volume e peso, cria um Agente novo que representara esse entregador, vinculando um numero de identificação. Apos a definição dos pacotes, o agente notifica o Centro e este vincula os dados aos entregadores, salvando a informação final.

Figure 3: Sistema de troca de mensagens

Como anotado na 3 o agente Centros também tem o papel de filtrar as informações que chegam aos agentes entregadores, tentando diminuir a informações desnecessárias ou pouco relevantes para cada individuo.

4 PRÓXIMAS ETAPAS

A arquitetura do sistema está em fase final de implementação. A mediação entre os entregadores viabilizada por coalizão contará com três algoritmos de diferentes, a saber: (i) *simulated annealing* [7], (ii) *market-based* [15] e (iii) e o clássico algoritmo de Shehory and Kraus [12].

A partir destes algoritmos, a intenção é definir um benchmark de modo a avaliar a adequação dos métodos baseados em características como demanda de tarefas, restrições de recursos, e função objetivo com a região de atuação do sistema. A intenção é entender em que condições cada método entrega os melhores resultados segundo as características indicadas acima.

5 CONCLUSÃO

Neste artigo foi apresentado uma arquitetura de sistema mediado por agentes para a solução do problema de roteamento de veículo em *last-mile*. O sistema trabalha com a hipótese de que VRP pode ser modelado como uma coalizão de pacotes de entregas a serem atribuídas a um conjunto de entregadores. Deste modo, é possível utilizar os diferentes algoritmos existentes na literatura de modo a reconhecer as características de cada região de distribuição que melhor de adequam a cada algoritmo disponível no sistema.

O desdobramento imediato desta sistema está no uso de técnicas de *ensemble* [6], onde se combinam diferentes modelos de predição para a assistência cada aspecto singular do problema e cobrir todos os comportamentos de um sistema de modo a levar um melhor resultado de predição e uma melhor performance. O principal objetivo no uso de modelos *ensemble* é simplificar os processos de coalizão e melhorar sua performance por meio da combinação de seus resultados.

REFERENCES

[1] Dariusz Barbucha. 2014. A Cooperative Agent-Based Multiple Neighborhood Search for the Capacitated Vehicle Routing Problem. *Advances in Intelligent Systems and Computing* 234 (01 2014), 129–143. https://doi.org/10.1007/978-3-319-01649-8_9

[2] Dariusz Barbucha and Piotr Jedrzejowicz. 2009. Agent-Based Approach to the Dynamic Vehicle Routing Problem, Vol. 55. 169–178. https://doi.org/10.1007/978-3-642-00487-2_18

[3] Kris Braekers, Katrien Ramaekers, and Inneke Van Nieuwenhuyse. 2016. The vehicle routing problem: State of the art classification and review. *Comput. Ind. Eng.* 99 (2016), 300–313.

[4] Burak Eksioglu, Arif Vural, and Arnold Reisman. 2009. The vehicle routing problem: A taxonomic review. *Computers Industrial Engineering* 57 (11 2009), 1472–1483. https://doi.org/10.1016/j.cie.2009.05.009

[5] Juan Camilo Fonseca Galindo, Gabriela de Castro Surita, José Maia Neto, Cristiano Leite de Castro, and André Paim Lemos. 2020. A Multi-Agent System for Solving the Dynamic Capacitated Vehicle Routing Problem with Stochastic Customers using Trajectory Data Mining. *CoRR* abs/2009.12691 (2020). arXiv:2009.12691 https://arxiv.org/abs/2009.12691

[6] Mehdi Golzadeh, Esmaeil Hadavandi, and S Chehreh Chelgani. 2018. A new Ensemble based multi-agent system for prediction problems: Case study of modeling coal free swelling index. *Applied Soft Computing* 64 (2018), 109–125.

[7] Helena Keinänen. 2009. Simulated annealing for multi-agent coalition formation. In *KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications*. Springer, 30–39.

[8] David Lai, Ozgun Caliskan-Demirag, and Janny Leung. 2016. A tabu search heuristic for the heterogeneous vehicle routing problem on a multigraph. *Transportation Research Part E: Logistics and Transportation Review* 86 (02 2016), 32–52. https://doi.org/10.1016/j.tre.2015.12.001

[9] Mingqi Lv, Ling Chen, Tieming Chen, Dajian Zeng, and Bin Cao. 2019. Discovering individual movement patterns from cell-id trajectory data by exploiting handoff features. *Information Sciences* 474 (feb 2019), 18–32. https://doi.org/10.1016/j.ins.2018.09.033

[10] Meng Qiu, Zhuo Fu, Richard Eglese, and Qiong Tang. 2018. A Tabu Search algorithm for the vehicle routing problem with discrete split deliveries and pickups. *Computers Operations Research* 100 (08 2018), 102–116. https://doi.org/10.1016/j.cor.2018.07.021

[11] Efrain Ruiz, Valeria Soto-Mendoza, Alvaro Barbosa, and Ricardo Reyes. 2019. Solving the Open Vehicle Routing Problem with Capacity and Distance Constraints with a Biased Random Key Genetic Algorithm. *Computers Industrial Engineering* 133 (05 2019). https://doi.org/10.1016/j.cie.2019.05.002

[12] Onn Shehory and Sarit Kraus. 1995. Task allocation via coalition formation among autonomous agents. In *IJCAI (1)*. Citeseer, 655–661.

[13] Onn M Shehory, Katia Sycara, and Somesh Jha. 1997. Multi-agent coordination through coalition formation. In *International Workshop on Agent Theories, Architectures, and Languages*. Springer, 143–154.

[14] Juan Antonio Sicilia, Carlos Quemada, Beatriz Royo, and David Escuín. 2016. An optimization algorithm for solving the rich vehicle routing problem based

on Variable Neighborhood Search and Tabu Search metaheuristics. *J. Comput. Appl. Math.* 291 (2016), 468–477. https://doi.org/10.1016/j.cam.2015.03.050 Mathematical Modeling and Computational Methods.

[15] Lovekesh Vig and Julie A Adams. 2006. Market-based multi-robot coalition formation. In *Distributed Autonomous Robotic Systems 7*. Springer, 227–236.

[16] Richard T. Wong. 2008. *Vehicle Routing for Small Package Delivery and Pickup Services*. Springer US, Boston, MA, 475–485. https://doi.org/10.1007/978-0-387-77778-8_21

[17] Michael Wooldridge. 2009. *An Introduction to Multiagent Systems* (2 ed.). Wiley, Chichester, UK.

Processos Markovianos de Decisão: uma análise comparativa

ALEXIA CAROLINA SCHEFFER DA SILVA and VALDINEI FREIRE DA SILVA, Universidade de São Paulo, Brazil

Para o problema de entrega de pacotes, pode-se adotar Processos Markovianos de Decisão (MDP), que é um dos arcabouços mais utilizado para problema de decisão em ambientes estocásticos. Entretanto, dado qualquer problema de modelagem, pode ser aplicado sobre diferentes funções valores, sendo o mais comumente adotado, o valor esperado de um custo acumulado. Esse projeto propõe trazer uma modelagem sobre o problema de entrega de pacotes em que aborda função valor além do fundamental, que busca o menor custo para atingir um objetivo e explora os possíveis impactos que tais definições podem trazer. Para isso, atenta-se à perspectiva da pessoa que recebe um pacote e se considera o tempo de espera para recebê-lo. Além disso, de forma a conciliar o potencial de alta dimensão do problema, buscou-se avaliar empiricamente com algoritmos de aprendizado por reforço.

Additional Key Words and Phrases: processos markovianos de decisão, caminho mais curto estocástico, planejamento probabilístico, iteração de valor, aprendizado por reforço, advantage actor critic, otimização, sensibilidade ao risco, rota ótima para entrega de pacotes

ACM Reference Format:
Alexia Carolina Scheffer da Silva and Valdinei Freire da Silva. 2021. Processos Markovianos de Decisão: uma análise comparativa. 1, 1 (December 2021), 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUÇÃO

Processos Markovianos de Decisão (*Markov Decision Process* – MDP) permitem modelar problemas de tomada de decisão em ambientes com respostas estocásticas, sendo um dos arcabouços mais utilizados para esse tipo de problema [Mausam and Kolobov 2012a]. Como exemplo, MDP é utilizado no paradigma de Aprendizado por Reforço, assim como em planejamento estocástico [Sutton and Barto 2018].

Um MDP descreve o processo de um agente atuando em um ambiente, no qual: (*i*) o agente observa o estado s_t do ambiente; (*ii*) escolhe e executa uma ação a_t; e (*iii*) o ambiente transita para um novo estado s_t. Objetivo do agente é encontrar uma política de ação $\pi : \mathcal{S} \rightarrow \mathcal{A}$, que mapeia estados em ação. Assim, uma política ótima maximiza alguma função valor V^π que avalia a qualidade de políticas arbitrárias.

Usualmente, MDPs consideram uma função custo $c : \mathcal{S} \times \mathcal{A} \rightarrow \Re$ que representa um custo imediato, enquanto a função valor V^π sumariza o custo acumulado, assim como a estocasticidade resultante da política atuando no ambiente. A função valor mais comum é simplesmente o valor esperado do custo acumulado, considerada neutro ao risco, mas várias formulações que levam em conta a sensibilidade ao risco podem ser encontradas na literatura [Chow et al. 2018; Howard and Matheson 1972; Mihatsch and Neuneier 2002; Minami and Silva 2012].

Logo, dependendo do problema, diferentes funções valores podem representar os anseios de diferentes atores no problema. Por exemplo, considere uma empresa que fornece o serviço de entregar pacotes. Enquanto o CEO da empresa pode ter o objetivo de diminuir os custos do processo e, por exemplo, minimizar a quantidade de quilômetros rodados por seus veículos, pela perspectiva dos clientes desses serviços, poderia ter o objetivo de minimizar o tempo de espera para que um pacote seja entregue. Esses objetivos, em geral, são conflitantes, por exemplo, a estratégia de acumular vários pacotes para fazer uma entrega conjunta percorrendo menos quilômetros, aumenta o tempo de espera para que um pacote seja entregue.

Quando o agente está sob poder de um único ator, este ator é responsável por determinar o balanço entre seus anseios e os anseios de outros atores afetados pelas decisões do agente planejador. Embora consideremos um ambiente com um único agente, do ponto de vista do ator responsável, os outros atores também são agentes e podem pressionar o ator responsável por seus anseios, por exemplo, deixando de contratar os serviços. Não é incomum que o equilíbrio escolhido pelo ator responsável trate os outros autores de acordo com seu poder de barganha, produzindo desigualdade sistemática nos anseios dos outros atores e aplicando ao caso de entrega de pacotes, tendo entregas com tempo de espera muito maior para grupos específicos. O caso recente da escolha de cobertura para entregas com frete gratuito pela Amazon[1] é possível observar esse reflexo.

Uma forma de ver o problema de equidade é restringir para que grupos de atores, agrupados por alguma variável sensível, receba o mesmo tratamento para os seus anseios. Algumas outras medidas flexibiliza a equidade, desde de que grandes ganhos sejam obtidos para outros atores, como por exemplo, a soma de potências dos custos individuais de cada ator, ou um compromisso entre custo médio total e variância entre os custos individuais [Kasy and Abebe 2021].

Embora alguns trabalhos em MDPs abordam critérios multiobjetivos, encontrando uma fronteira de soluções não dominadas, no caso em que temos vários atores, a dimensionalidade pode ser proibitiva. Enquanto os algoritmos de planejamento tem sofrido com o mal da dimensionalidade, algoritmos de Aprendizado por Reforço tem sido aplicados de forma bem sucedida para problemas com alta dimensão.

Esse trabalho tem como objetivo avaliar empiricamente a capacidade de algoritmos de aprendizado por reforço para encontrar equilíbrios arbitrários entre multiobjetivos. Os experimentos são realizados em ambientes pequenos, para os quais é possível o planejamento, para que se possa comparar as soluções obtidas via planejamento e aprendizado por reforço.

Authors' address: Alexia Carolina Scheffer da Silva, ac.scheffer@usp.br; Valdinei Freire da Silva, valdinei.freire@usp.br, Universidade de São Paulo, Rua Arlindo Béttio, 1000, São Paulo, SP, Brazil, 03828-000.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2021 Association for Computing Machinery.
XXXX-XXXX/2021/12-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

[1] https://www.bloomberg.com/graphics/2016-amazon-same-day/

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Processos Markovianos de Decisão

Um processo markoviano de decisão [Mausam and Kolobov 2012b] é definido por uma tupla $\langle \mathcal{S}, \mathcal{A}, T, c, H \rangle$ em que:

- $s \in \mathcal{S}$ são estados possíveis;
- $a \in \mathcal{A}$ são ações possíveis;
- $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ é a função de transição;
- $c : \mathcal{S} \times \mathcal{A} \to \mathfrak{R}$ é a função recompensa; e
- H é o horizonte para tomada de decisão.

Um MDP define o seguinte processo para todo tempo discreto $t \in \{0, 1, \ldots, H-1\}$:

(1) o agente percebe o estado s_t;
(2) o agente escolhe e executa uma ação a_t;
(3) o ambiente transita para o próximo estado s_{t+1} com probabilidade $\Pr(s_{t+1} = s' | a_t = a, s_t = s) = T(s, a, s')$; e
(4) o agente paga um custo $c(s_t, a_t)$.

Usualmente, quando o horizonte H é finito, considera-se que um agente age de forma ótima se este agente minimiza o custo acumulado esperado. Nesse caso, a solução ótima é descrita por uma política não-estacionária $\pi : \mathcal{S} \times \mathbb{N}$ tal que $a_t = \pi(s_t, t)$ e o valor de uma política é definido por:

$$V^{\pi} = \mathrm{E}\left[\sum_{t=0}^{H-1} c_t \middle| \pi \right].$$

2.2 Algoritmo de Iteração de Valor

Um algoritmo tradicional para encontrar uma política ótima é o algoritmo Iteração de Valor. Esse algoritmo considera uma função valor $V : \mathcal{S} \times \mathbb{N} \to \mathfrak{R}$, na qual $V(s, t)$ define o custo acumulado a partir do estado t quando o processo está no estado s, isto é:

$$V^*(s, t) = \mathrm{E}\left[\sum_{i=t}^{H-1} c_i \middle| s_0 = s, \pi^* \right].$$

O algoritmo Iteração de Valor pode ser visto na Tabela 1 e se baseia na equação de Bellman:

$$V^*(s, t) = \min_{a \in \mathcal{A}} \left\{ c(s, a) + \sum_{s' \in \mathcal{S}} T(s, a, s') V^*(s', t+1) \right\}.$$

(1) defina $V^*(s, H) = 0$
(2) faça para todo passo $t = H - 1$ to 0
  (a) para todo $s \in \mathcal{S}$

$$V^*(s, t) = \min_{a \in \mathcal{A}} \left\{ c(s, a) + \sum_{s' \in \mathcal{S}} T(s, a, s') V^*(s', t+1) \right\}$$

  (b) para todo $s \in \mathcal{S}$

$$\pi(s, t) = \arg\min_{a \in \mathcal{A}} \left\{ c(s, a) + \sum_{s' \in \mathcal{S}} T(s, a, s') V^*(s', t+1) \right\}$$

(3) retorne π

Table 1. Algoritmo Iteração de Valor

2.3 *Advantage Actor-Critic*

De acordo com o artigo de Chris Yoon[2], em algoritmos com derivação de gradiente de política convencionais, como em Monte Carlo, ocorrem problemas de gradientes barulhentos e alta variância, tornando-se instáveis e de convergência lenta a métodos de gradientes de política. Assim, uma forma de estabilizar e encontrar convergência no algoritmo, é subtraindo a recompensa cumulativa através de uma linha de base, criando, portanto, gradientes e atualizações menores e mais estáveis. A linha de base pode assumir vários valores, dentre eles, está o *Advanced Actor Critic*.

Algoritmos que atualizam a política antes que ela seja completamente avaliada implementam a Iteração de Política Generalizada (GPI)[Szepesvári 2019]. No GPI, existem dois processos em estreita interação:

- ***Critic***, em que avalia a política atual e se estima a função de valor, podendo ser valor da ação ou valor do estado.
- ***Actor***, em que atualiza a distribuição da política na direção sugerida de *Critic* e visa melhorar a política atual.

Ambas funções podem ser parametrizadas com redes neurais. Sendo assim, pode-se determinar o *advantage value*, em que toma como argumentos a função valor de estado para a *baseline* adotada, tratando-se do quão melhor é a ação específica em relação a ação geral em determinado estado.

A política que é usada para gerar as amostras (política de comportamento) pode ser diferente daquela que é avaliada e melhorada no sistema Actor-Critic (política de destino). Isso pode ser útil porque o Critic deve aprender sobre as ações não preferidas pela política de metas atual para que possa melhorá-la. Isso é impossível de se conseguir se a política de comportamento for igual à política de destino e se a política de destino for determinística. Esse é um dos motivos pelos quais a política de destino geralmente é uma política estocástica. No entanto, mesmo que a política de destino seja estocástica, a qualidade das estimativas dos valores das ações de baixa probabilidade pode ser muito pobre, uma vez que menos informação é recebida para tais ações. Pode parecer então que escolher ações completamente aleatórias pode fornecer mais informações. No entanto, esse claramente não é o caso, uma vez que tal política aleatória pode não visitar as partes importantes do espaço de estados, como discutido antes. Portanto, na prática, a política de comportamento geralmente combina uma certa (pequena) quantidade de exploração na política de destino[Szepesvári 2019].

Existem muitas maneiras de implementar uma arquitetura Actor-Critic. Se o espaço de ação for pequeno, o Critic pode, por exemplo, usar uma função de valor de ação aproximada e o Actor pode seguir uma estratégia de exploração ϵ-gulosa ou de Boltzmann. Se o espaço de ação for grande ou contínuo, o próprio Actor pode usar a função de aproximação.[Szepesvári 2019] Observe que, ao contrário da Iteração de Política perfeita, um método GPI pode gerar uma política substancialmente pior do que a anterior. Assim, a qualidade da sequência de políticas geradas pode oscilar ou mesmo divergir quando

[2]https://towardsdatascience.com/understanding-actor-critic-methods-931b97b6df3f

a etapa de avaliação da política está incompleta, independentemente de a melhoria da política ser exata ou aproximada. Na prática, o GPI tende a gerar políticas que melhoram no início. No entanto, em estágios posteriores, as políticas costumam oscilar. Uma prática comum, portanto, é armazenar a sequência de políticas obtidas e, quando o aprendizado terminar, medir o desempenho das políticas armazenadas executando alguns testes e, a seguir, selecionar aquele empiricamente com melhor desempenho.[Szepesvári 2019] Assim como no caso da iteração de valor ajustado, o desempenho dos métodos Actor-Critic pode ser controlado aumentando a "flexibilidade" dos métodos de aproximação de função.

2.4 Mensurando Iniquidade

Considere que em um MDP, um único agente atua no ambiente tomando decisões, mas que essas decisões afetam diferentes atores, isto é, cada ator $p \in \mathcal{P}$ possui uma função custo $c_p : \mathcal{S} \times \mathcal{A} \to \mathfrak{R}$. Dessa forma, para cada ator p e política π pode-se definir a função valor:

$$V^{p,\pi} = \mathrm{E}\left[\sum_{t=0}^{H-1} c^p(s_t, a_t) \middle| \pi\right].$$

Uma política garante equidade se todos os atores possuem a mesma função valor, isto é, para cada par $p, q \in \mathcal{P}$, tem-se que $V^{p,\pi} = V^{q,\pi}$.

Embora a equidade seja desejável, se o ambiente não apresenta simetria com relação aos atores, pode ser que o problema não seja bem posto se o objetivo seja garantir a equidade.

Pode-se considerar duas opções relaxadas para garantir equidade: *worst case* e *measure of welfare* [Kasy and Abebe 2021]. No primeiro caso considera-se o ator com pior retorno, isto é, o valor de uma política é dado por:

$$V^{\pi} = \max_{p \in \mathcal{P}} V^{p,\pi}.$$

No segundo caso considera-se uma função utilidade $u : \mathfrak{R} \to \mathfrak{R}$ que modula o custo e considera-se a utilidade total para todos os atores, isto é:

$$V^{\pi} = \sum_{p \in \mathcal{P}} \mathrm{E}\left[u\left(\sum_{t=0}^{H-1} c^p(s_t, a_t)\right) \middle| \pi\right].$$

Essa avaliação de política é aversa a iniquidade se a função utilidade u é aversa ao risco, por exemplo, $u(x) = x^{\lambda}$, na qual λ parametriza aversão a iniquidade, quanto maior, mais aversão.

2.5 Otimizando equidade em MDPs

Ao considerar planejamento de aversão a iniquidade em MDPs, pode-se levar em conta aquela que mais facilmente se otimiza. Sabe-se que o algoritmo de Iteração de Valor pode ser utilizado para maximizar a esperança de qualquer função utilidade sobre custos.

Para otimizar a esperança de uma função utilidade u arbitrária deve-se considerar duas extensões em MDPs. Primeiro, deve-se estender o espaço de estados com o custo acumulado até o tempo t, isto é, defini-se um espaço de estados $\mathcal{X} = \mathcal{S} \times \mathfrak{R}$ e o estado estendido é dado por $x_t = \left(s_t, \sum_{i=0}^{t-1} c_t\right)$. Segundo, defini-se uma nova função custo $c^u : \mathcal{X} \times \mathcal{A} \to \mathfrak{R}$ tal que:

$$c^u(x = (s, C), a) = u(C + c(s, a)) - u(C);$$

garantindo que $\sum_{t=0}^{H-1} c_t^u = u\left(\sum_{t=0}^{H-1} c(s_t, a_t)\right)$ [Freire and Delgado 2017].

No caso em que a função utilidade depende de várias funções utilidades, como na aversão a iniquidade, o espaço de estados deve ser estendido com várias dimensões trazendo a tona a maldição da dimensionalidade.

3 O PROBLEMA DE DESPACHO DINÂMICO

Neste artigo considera-se o problema de despacho dinâmico de pacotes que considera o seguinte processo:

(1) Os pacotes chegam em fila em um centro de distribuição localizado na coordenada x, y;
(2) cada pacote i possui como destino coordenadas x_i, y_i;
(3) os pacotes devem ser distribuídos em ordem de chegada a um dos veículos disponíveis, isto é, que não estão cheios;
(4) em qualquer momento, qualquer um dos veículos podem ser despachados;
(5) ao ser despachado, os veículos percorrem uma rota ótima para realizar a entrega dos pacotes a cada um dos destinos de cada pacote no veículo e retorna ao centro de distribuição; e
(6) se todos veículos estão cheios, algum veículo deve ser despachado para que um novo veículo ocupe sua posição e o pacote atual possa ser distribuído.

3.1 Formalismo para MDPs

Para formalizar o problema de despacho dinâmico como um MDP, precisa-se definir o espaço de estados, o espaço de ações, a função de transição e a função de custo.

Primeiro, considera-se que a todo momento há N_v veículos no centro de distribuição e sempre que um veículo sai para fazer entrega, outro veículo ocupa o lugar do mesmo.

O estado do ambiente é definido por:

- as coordenadas de destino do pacote para ser distribuído; e
- as coordenadas de cada pacote em cada carro.

O espaço de ações é definido por:

- para qual veículo o pacote atual será distribuído; e
- para cada veículo se ele será despachado ou não.

A função de transição responde deterministicamente às ações, isto é, ao pacote atual que é alocado no veículo escolhido e aos veículos despachados que fazem a entrega. A única fonte de estocasticidade é o destino do próximo pacote, que pode ser modelado por uma função densidade de probabilidade sobre o espaço de coordenadas.

Considera-se dois tipos de custo: espera e distância percorrida. O custo de espera é associado a cada pacote de entrega e é unitário a cada passo de tempo que o pacote fica no centro de distribuição[3]. Já, o custo de distância é a distância percorrida acumulada por todos veículos que foram despachados naquele passo de tempo.

Como tem-se vários objetivos conflitantes, utiliza-se a função utilidade $u(x) = x^2$ para modelar aversão a iniquidade. Como espera e distância são de dimensões e poderes diferentes, são levantados fatores α e $1-\alpha$ que ponderam a espera e distância, respectivamente.

[3]Como simplificação, desconsiderou-se o tempo decorrido durante a entrega.

3.2 Modelo Simplificado

O modelo definido anteriormente apresenta várias dificuldades para se obter políticas ótimas. Primeiro, o espaço de estado tem muitas dimensões que aumentam exponencialmente com a quantidade de veículos e a quantidade de espaço disponível em cada veículo. Segundo, se qualquer coordenada no espaço dos reais é possível, temos um estado contínuo. Terceiro, a quantidade de pacotes é potencialmente infinita. Quarto, se aversão a iniquidade é considerada, o espaço de estado deve ser estendido para cada pacote.

Na seção de experimentos considerou-se um modelo simplificado com os seguintes parâmetros:

- a quantidade de veículos foi limitada a apenas dois veículos, com apenas dois espaços disponíveis;
- a quantidade de coordenadas para entrega foi limita a apenas duas localidades, facilitando o planejamento da rota ótima, assim como tornando o problema discreto;
- o tempo máximo de espera de cada objeto é no máximo 5 passos de tempo, forçando o despacho do carro com este objeto quando este limite é atingido; e
- a quantidade máxima de objetos despachados é limitada por 10.

Mesmo com todas essas simplificações, o espaço de estados estendidos tem cardinalidade de 2.292.000 estados.

O espaço de estados também foi reduzido para quatro ações:

- distribuir o pacote atual para o veículo 1, caso o veículo esteja cheio, o veículo é despachado e o pacote é distribuído no novo veículo;
- distribuir o pacote atual para o veículo 2, caso o veículo esteja cheio, o veículo é despachado e o pacote é distribuído no novo veículo;
- despachar o veículo 1 e o pacote é distribuído no novo veículo; e
- despachar o veículo 2 e o pacote é distribuído no novo veículo.

4 RESULTADOS

Para realizar os experimentos, considerou-se o ambiente simplificado descrito na seção 3.2. O centro de distribuição foi alocado nas coordenadas $(0, 0)$, a localidade 1 foi alocada nas coordenadas $(0, -2)$ e a localidade 2 foi alocada nas coordenadas $(0, 0.5)$. Note que não há vantagem em fazer entregas para as duas localidades em conjunto.

Foram avaliadas 3 distribuições de probabilidades para as localidades do destino dos pacotes: balanceado com ambas localidades com probabilidade 0.5; desbalanceado em favor da localidade 1 com probabilidade 0.8; e desbalanceado em favor da localidade 2 com probabilidade 0.8. Os resultados são mostrados respectivamente nas Figuras 1, 2 e 3.

As figuras mostram o compromisso obtido para vários valores de α entre: a distância percorrida, a espera média das entregas na localidade 1, e a espera média das entregas na localidade 2. Os valores de α foram obtidos considerando uma abordagem geométrica de construção da fronteira não dominada [Freire da Silva and Reali Costa 2011].

Na maioria dos cenários e compromissos, a localidade 1 está em desvantagem, sendo prejudicada por conta de estar a uma distância maior do centro de distribuição. A maior diferença ocorre também quando há um desbalanceamento em favor da localidade 2 (Figura 3).

Fig. 1. Quantidades de objetos nas Localidade 1 e Localidade 2 balanceadas.

Fig. 2. Quantidades de objetos nas Localidade 1 e Localidade 2 desbalanceadas em favor da Localidade 1.

Mesmo quando a localidade 1 possui mais pacotes, a localidade 1 só apresenta vantagem quando o compromisso dá maior peso para a distância percorrida (Figura 2).

O algoritmo A2C detalhado na 2.3 está como experimento em andamento e para ser finalizado. O algoritmo está implementado considerando o modelo simplificado e têm como próximas etapas a coleta de resultados e a análise comparativa.

5 CONCLUSÃO

O objetivo desse projeto era avaliar empiricamente a capacidade de algoritmos de aprendizado por reforço para encontrar equilíbrios arbitrários entre multiobjetivos, além de realizar a aplicação desses

Fig. 3. Quantidades de objetos nas Localidade 1 e Localidade 2 desbalanceadas em favor da Localidade 2.

algoritmos em um contexto factível, como exemplo próximo disso, o repositório da LoggiBUD.

Para isso, houve a exploração teórica apresentada previamente para, consoante aos estudos, aplicar e implementar os algoritmos possíveis, dado o período estipulado.

Devido a falta de familiarização ao tema e a complexidade da área de estudo, um grande período foi dedicado a estudar conceitos fundamentais para a pesquisa e assim, o número de algoritmos implementados para fins de comparação e análise de resultados foi reduzido. Como também, a maior parte da pesquisa foi executada modelo simples, da qual tinha como proposta final a aplicação no repositório LoggiBUD.

6 TRABALHOS FUTUROS

Levando em consideração toda a exploração teórica realizada e os resultados obtidos ao longo do processo de pesquisa, é possível ser levantado como contribuições futuras ao projeto:

- Finalização da implementação do algoritmo A2C para coleta de resultados e avaliação comparativa em relação ao planejamento para verificar se os dados são coerentes.
- Implementação de mais algoritmos de planejamento probabilístico e de aprendizado por reforço ao modelo simples para análise e comparação de resultados.
- Modelagem de novos conjuntos de estados e de ações para adquirir resultados mais consistentes ao aplicar nos algoritmos e aprimorar a aplicação do código ao contexto da realidade do problema.
- Adaptação do código implementado e aplicação dos algoritmos testados ao repositório LoggiBUD para demonstração dos resultados obtidos dado cada algoritmo.

ACKNOWLEDGMENTS

Este trabalho foi realizado com o apoio da Loggi , por meio do Programa de Bolsas de Pesquisa Loggi (PBP-LOGGI).

REFERENCES

Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. 2018. Risk-Constrained Reinforcement Learning with Percentile Risk Criteria. *Journal of Machine Learning Research* 18, 167 (2018), 1–51. http://jmlr.org/papers/v18/15-636.html

Valdinei Freire and Karina Valdivia Delgado. 2017. GUBS: a Utility-Based Semantic for Goal-Directed Markov Decision Processes. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. 741–749.

Valdinei Freire da Silva and Anna Helena Reali Costa. 2011. A Geometric Approach to Find Nondominated Policies to Imprecise Reward MDPs. In *Machine Learning and Knowledge Discovery in Databases*, Dimitrios Gunopulos, Thomas Hofmann, Donato Malerba, and Michalis Vazirgiannis (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 439–454.

Ronald A. Howard and James E. Matheson. 1972. Risk-sensitive Markov decision processes. *Management Science* 18, 7 (1972), 356–369.

Maximilian Kasy and Rediet Abebe. 2021. Fairness, Equality, and Power in Algorithmic Decision-Making. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency* (Virtual Event, Canada) *(FAccT '21)*. Association for Computing Machinery, New York, NY, USA, 576–586. https://doi.org/10.1145/3442188.3445919

Loggi. 2021. loggiBUD: Loggi Benchmark for Urban Deliveries. https://github.com/loggi/loggibud. *GitHub repository* (2021).

Mausam and Andrey Kolobov. 2012a. Planning with Markov Decision Processes: An AI Perspective. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6, 1 (2012), 1–210. https://doi.org/10.2200/S00426ED1V01Y201206AIM017 arXiv:https://doi.org/10.2200/S00426ED1V01Y201206AIM017

Mausam and Andrey Kolobov. 2012b. *Planning with Markov Decision Processes: An AI Perspective*. Morgan Claypool Publishers. http://dx.doi.org/10.2200/S00426ED1V01Y201206AIM017

O. Mihatsch and R. Neuneier. 2002. Risk-Sensitive Reinforcement Learning. *Machine Learning* 49 (2002), 267–290.

Renato Minami and Valdinei Freire da Silva. 2012. Shortest stochastic path with risk sensitive evaluation. In *11th Mexican International Conference on Artificial Intelligence (MICAI 2012) (Lecture Notes in Artificial Intelligence, Vol. 7629)*. 370–381.

Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction* (second ed.). The MIT Press.

Richard S. Sutton and Andrew G. Barto. 2020. *Reinforcement Learning: An Introduction* (second ed.). The MIT Press. http://incompleteideas.net/book/the-book-2nd.html

Csaba Szepesvári. 2019. *Algorithms for Reinforcement Learning*. Morgan Claypool Publishers. http://dx.doi.org/10.2200/S00268ED1V01Y201005AIM009

Loggi Hub Point (2021)

Juliana Aragão Pinto*
J.A Pinto
aragaopintojuli@gmail.com
Universidade Estadual de Feira de Santana
Feira de Santana, Bahia, BR

João B. Rocha-Junior
Rocha-Junior, J. B
joao@uefs.com
Universidade Estadual de Feira de Santana
Feira de Santana, Bahia, BR

Abstract

The location of a new Hub (nodal point) has impacts on the cost and the quality of the deliveries. Moreover, it is fundamental to better employ the qualities of a multimodal transport system. On adding a new Hub, there are several objectives that can be pursued such as: 1) balancing the number of packages to be delivered by each Hub or 2) balancing the distance needed to deliver the packages by each Hub. In this paper, we present two algorithms to selet the k best hub candidates, from a set of new Hub candidates, in order to minimize the total distance (MinSum) and to minimize the maxdistace (MinMax) required to deliver the packages.

Keywords: logistics, optimal location queries,application

ACM Reference Format:
Juliana Aragão Pinto, J.A Pinto, João B. Rocha-Junior, and Rocha-Junior, J. B. 2021. Loggi Hub Point (2021). In *Proceedings of Loggi Workshop (PBP-Loggi).* PBP-Loggi, BR, 6 pages.

1 Introdução

O Loggi Leve[1] é o sistema utilizado pela Loggi para ampliar a distribuição de pacotes em grandes e pequenas cidades. O Loggi Leve permite que a Loggi utilize o serviço de transportadoras já existentes (ou novas) para ampliar a entrega de pacotes. Para se tornar um franqueado e participar da distribuição de pacotes da Loggi, uma transportadora preenche um cadastro, que entre outros dados, informa o endereço do espaço físico que será utilizado para comportar os pacotes antes de serem distribuídos para os clientes. Um Hub (*nodal point*), neste projeto, é um local físico onde os pacotes ficam armazenados por uma transportadora (Loggi Leve) antes de serem distribuídos para os clientes.

Um Hub é um centro de distribuição [Krzyżaniak et al. 2012] de pacotes de pequeno porte. Os pacotes distribuídos pelo Hub podem variar desde um envelope até uma caixa que pode ser transportada em um carro de pequeno porte. O Hub é o último ponto de conexão entre o pacote e o cliente final, a maioria dos pacotes saem dos Hubs em veículos de pequeno porte como bicicletas ou motocicletas para serem entregues aos clientes.

A decisão sobre adicionar ou não um novo Hub no sistema deve levar em consideração diversos fatores, entre eles a localização deste novo Hub. O local deste novo Hub tem impacto sobre o custo e a qualidade da distribuição de pacotes. Por exemplo, colocar um novo Hub em um local que ainda não é atendido pelos Hubs existentes, permite reduzir o deslocamento médio total necessário para entregar os pacotes, impactando positivamente sobre o tempo total e na qualidade do serviço. Entretanto, o novo Hub pode impactar sobre as entregas dos Hubs existentes, diminuindo o número de pacotes destinados aos Hubs que atendiam previamente uma dada região. Isso pode impactar sobre o faturamento das franquias (Loggi Leve) já existentes. Portanto, definir o local de um novo Hub é uma tarefa complexa e que precisa ser analisada sobre diversos aspectos.

A Consulta por Localização Ideal (*Optimal Location Queries*) [Bin Yao 2011; Zitong Chen 2015] pode ser utilizada para fornecer informações e auxiliar na tomada de decisão da Loggi sobre o local de um novo Hub. Dado um conjunto de locais de destino dos pacotes D, um conjunto de Hubs existentes H e um conjunto de locais candidatos para instalar um novo Hub C, a Consulta por Localização Ideal retorna o melhor local candidate $c \in C$ que maximiza uma função de custo.

Existem duas funções de custo principais: MinSum e MinMax. A função MinSum visa minimizar a soma das distâncias percorridas para entregar os pacotes (tendo impacto sobre o tempo e o custo para realizar as entregas). Enquanto que a função de custo MinMax, visa minimizar a distância máxima entre todos os hubs, incluindo o hub candidato, e os locais de destino dos pacotes (tendo impacto sobre o tempo e o custo para realizar as entregas).

O objetivo deste artigo é apresentar as atividades desenvolvidas e resultados parciais obtidos durante o primeiro ano do projeto de pesquisa Hub Point. Neste primeiro ano, nós implementamos os algoritmos MinSum e MinMax, utilizando a distância Euclidiana, e fizemos uma aplicação, utilizando

*Both authors contributed equally to this research.
[1]https://www.loggi.com/leve

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
PBP-Loggi, December 2021, BR
© 2021 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.

bases de dados fornecidas pela Loggi. A aplicação permite posicionar e selecionar os melhores Hubs Candidatos.

O restante deste artigo está organizado da seguinte forma: a Seção 2 apresenta os principais artigos consultados no desenvolvimento desta pesquisa. A Seção 3 apresenta os algoritmos, 3.1 MinSum e 3.2 MinMax. A Seção 4 apresenta a Aplicação desenvolvida para exibir posicionar locais candidatos e exibir os melhores locais de acordo com os algoritmos desenvolvidos. A Seção 5 contem as considerações finais.

2 Trabalhos relacionados

Donghui Zhang [2006] foi o primeiro a propor a Consulta por Localização ideal. Os algoritmos propostos utilizam a distância Euclidiana. Posteriormente, Bin Yao [2011] e Zitong Chen [2015] apresentaram algoritmos para processar a Consulta por Localização Ideal em redes de estradas, onde a medida de distância é obtida através do menor caminho. Todos estes trabalhos não tem como foco a área de logística, mas sim a localização de um novo estabelecimento que fique mais próximo dos clientes.

Sarnath Ramnath [2020] aborda a distribuição de demanda para um conjunto de centros de distribuição. Eles introduzem alguns pontos interessantes que não são abordados em outros trabalhos, como a capacidade de um centro de distribuição. Entretanto, o foco do trabalho é no balanceamento de carga entre os centros e não na análise do sistema ao adicionar um novo nó.

Krzyżaniak et al. [2012] apresenta os principais tipos de Centros de Distribuição e identifica alguns pontos que precisam ser considerados ao instalar um novo Centro de Distribuição, como sua vizinhança. Diferentemente, o nosso foco é na identificação dos melhores locais para posicionar um novo Hub, levando em consideração o histórico de entregas de pacotes e a posição dos Hubs atuais.

A maioria dos trabalhos [Carolina Luisa dos Santos Vieira 2016; Senay Oguztimur 2018] focam na localização de grandes centros de distribuição, levando em consideração recursos naturais, benefícios sociais, o preço do imóvel, e a proximidade em relação a portos e aeroportos. Portanto, possuem objetivos diferentes do proposto neste artigo.

3 Algoritmos

Esta seção apresenta os algoritmos para processar duas consultas de localização ideal: MinSum (Seção 3.1) e MinMax (Seção 3.2).

3.1 MinSum

Dado um conjunto de deliveries (histórico de locais de entrega) D, um conjuntos de pontos de distribuição (hubs) existentes H e um conjunto de locais candidates a serem um novo hub C, o algoritmo MinSum (Algoritmo 1) retorna os

Algorithm 1: MinSum

input : A set of destinations D, a set hubs H and a set of hub candidates C
output : The k best candidates with the smallest sum distance

```
1  M ← MaxHeap();
2  foreach c ∈ C do
3      sum ← 0;
4      foreach d ∈ D do
5          minDist ← ∞;
6          foreach h ∈ H + c do
7              if dist(h,d) < minDist then
8                  minDist ← dist(h,d);
9              end
10         end
11         sum ← minDist + sum;
12     end
13     M.add((c, sum));
14     if |Heap| > k then
15         Heap.pop();
16     end
17 end
18 return M
```

k melhores candidatos que permitem reduzir a soma das distâncias utilizadas para entregar as encomendas.

O Algoritmo 1 apresenta o algoritmo MinSum. O algoritmo utiliza uma MaxHeap para manter os k melhores candidatos (linha 1), ou seja, os candidates que possuem a menor soma das distâncias. Para cada candidato $c \in C$, o algoritmo verifica se a soma das distâncias, utilizando este candidato é menor do que a soma das distâncias obtidas com outros candidatos. Este procedimento é feito adicionando o candidato a MaxHeap e mantendo a heap apenas com os k melhores candidatos (linhas 13-16). Para isto, o algoritmo precisa calcular a soma das distâncias para cada candidato (linhas 4-12). Este processamento é feito da seguinte forma: para cada local de entrega ($d \in D$), o algoritmo computa a menor distância entre d e o conjunto atual de hubs (H), assumindo que o candidato c a ser inserido faz parte deste conjunto (linhas 6-10). A menor distância (minDist) é então somada (linha 11) para obter as somas das menores distâncias para cada candidato c.

As figuras 1-4 ilustram o funcionamento deste algoritmo. A Figura 1 apresenta os locais de entrega (deliveries) d_i e dos hubs atuais H_i. A figura também apresenta os deliveries que são atraídos (pela menor distância) para cada hub. Por exemplo, os deliveries d_2, d_3, d_4 e d_6 são atraídos atualmente pelo hub H_1. Neste cenário, a soma das distâncias é soma das distâncias entre os deliveries $d_1 - d_8$ e os hubs H_1 e H_2, que neste caso é 39.

Figure 1. Hubs atuais

Figure 2. Candidatos

Figure 3. MinSum Candidato 1

Figure 4. MinSum Candidato 2

A Figura 2 apresenta o local dos dois candidatos a serem avaliados C_1 e C_2. Ao considerarmos a adição do hub candidato C_1 (Figura 3), a soma das distâncias entre os deliveries $d_1 - d_8$, os hubs H_1 e H_2 e o hub candidato C_1 é 28. Ao considerarmos a adição do hub candidato C_3 (Figura 4, a soma das distâncias entre os deliveries $d_1 - d_8$, os hubs H_1 e H_2 e o hub candidato C_2 é 27. Neste caso, considerando apenas os candidatos C_1 e C_2, o candidato C_2 possui uma soma das distâncias menor que C_1 e seria então o local candidato top-1 retornado pelo algoritmo.

3.2 MinMax

Dado um conjunto de deliveries (locais de entrega) D, um conjuntos de pontos de distribuição (hubs) existentes H e um conjunto de locais candidates a serem um novo hub C, o algoritmo MinMax (Algoritmo 2) retorna os k melhores candidatos que reduzem a distância máxima do conjunto total de deliveries.

O Algoritmo 2 apresenta o algoritmo MinMax. O algoritmo utiliza uma MinHeap para manter os k melhores candidatos (linha 1), ou seja, os candidates que possuem as menores distâncias máximas. Para cada candidato $c \in C$, o o algoritmo verifica se a maxDist, utilizando este candidato é menor do que a maxDist obtidas com outros candidatos. Este procedimento é feito adicionando o candidato a MaxHeap e mantendo a heap apenas com os k melhores candidatos (linhas 15-18). Para isto, o algoritmo precisa calcular a maior distância para cada candidato (linhas 4-12). Este processamento é feito da seguinte forma: para cada local de entrega ($d \in D$), o algoritmo computa a menor distância entre d e o conjunto atual de hubs (H), assumindo que o candidato c a ser inserido faz parte deste conjunto (linhas 6-10). A menor distância (minDist) é então verificada (linha 11) e caso ela seja menor que maxDist, a menor distância máxima é atualizada (linha 12).

As figuras 5-8 ilustram o funcionamento deste algoritmo. A Figura 5 apresenta os locais de entrega d_i e dos hubs atuais H_i. A figura também apresenta os deliveries que são atraídos (pela menor distância) para cada hub. Por exemplo, os deliveries d_2, d_3, d_4 e d_6 são atraídos atualmente pelo hub H_1. Nesse contexto, a maior distância entre os deliveries $d_1 - d_8$ e os hubs H_1 e H_2, vai ser o delivery H_2 d_1 de tamanho 12.

A Figura 6 apresenta o local dos dois candidatos a serem avaliados C_1 e C_2. Ao considerarmos a adição do hub candidato C_1 (Figura 3, a maior distância entre os deliveries $d_1 - d_8$, os hubs H_1 e H_2 e o hub candidato C_1 é 8. Ao considerarmos a adição do hub candidato C_3 (Figura 4), a maior distância entre os deliveries $d_1 - d_8$, os hubs H_1 e H_2 não muda do cenário contendo somente os Hubs Atuais (Figura 6), sendo assim, a maior distância entre do hub candidato C_2 é 12. Neste caso, considerando apenas os candidatos C_1 e C_2, o candidato C_1 possui o menor minmax e seria então o local candidato top-1 retornado pelo algoritmo.

Figure 5. Hubs atuais

Figure 6. Candidatos

Figure 7. MinMax Candidato 1

Figure 8. MinMax Candidato 2

4 Aplicação

O repositório Loggi Benchmark for Urban Deliveries (Loggibud)[2] foi utilizado no desenvolvimento da aplicação. O Loggibud oferece algumas funções prontas e dados que foram utilizados pela aplicação. Um exemplo disto são as funções para calcular das distâncias e as classes para lidar com os tipos de dados disponíveis na base de dados disponibilizada. As bases de dados contem informações sobre os locais de entrega (deliveries) e os Hubs atuais de alguns estados como Rio de Janeiro e Brasília.

A aplicação é dividida em duas partes principais: backend (Seção 4.1) e frontend (Seção 4.2). O backend contém uma *API* para o processamento das consultas MinMax e MinSum e para coletar dados que são exibidos pela aplicação. O frontend da aplicação contém um mapa interativo que permite selecionar as regiões, visualizar a localização dos deliveries (histórico de entrega) e dos hubs, posicionar hubs candidatos e exibir o resultado do processamento da consulta, destacando os melhores candidatos para cada cenário.

4.1 Backend

O Backend está hospedado em uma máquina virtual na *Google Cloud*[3]. Ele disponibiliza uma API desenvolvida em *Flask*[4] que fornece quatro rotas de acesso HTTPs que são utilizadas pela Aplicação:

- **[GET] /get-algorithms:** essa rota é responsável por retornar um vetor com os algoritmos desenvolvidos. Esses algoritmos são exibidos pela Aplicação e podem ser selecionados pelo usuário. Até o momento existe dois algoritmos disponíveis: MinMax e MinSum;
- **[GET] /get-regions:** essa rota retorna a localização e uma identificação (id) das regiões disponíveis na aplicação. Só estão disponíveis as regiões cujo as bases de dados foram disponibilizadas no Loggibud, que são: Brasília (DF), Rio de Janeiro (RJ) e Belém (PA);
- **[GET] /get-region-data:** essa rota retorna os locais dos hubs e os deliveries de uma dada região;
- **[POST] /run:** essa rota recebe como parâmetro o nomo algoritmo (MinMax ou MinSum), os locais dos hubs candidatos, o id da cidade e retorna o resultado dos algoritmos para os dados fornecidos, indicando os melhores candidatos.

4.2 Frontend

A Aplicação foi desenvolvida em React[5] e disponibilizada na nuvem da Vercel[6]. As principais funcionalidades da aplicação

[2]https://github.com/loggi/loggibud
[3]https://cloud.google.com
[4]https://flask.palletsprojects.com
[5]https://reactjs.org/
[6]https://loggiolq.vercel.app

Algorithm 2: MinMax

```
input  : A set of destinations D, a set hubs H and a
         set of hub candidates C
output : The k best candidates with the smallest max
         distance
M ← MinHeap();
foreach c ∈ C do
    maxDist ← 0;
    foreach d ∈ D do
        minDist ← ∞;
        foreach h ∈ H + c do
            if dist(h,d) < minDist then
                minDist ← dist(h,d);
            end
        end
        if minDist > maxDist then
            maxDist ← minDist;
        end
    end
    M.add((c,maxDist));
    if |Heap| > k then
        Heap.pop();
    end
end
R = [];
while M ≠ ∅ do
    R ← M.pop() + R;
end
return R;
```

são: 1) seleção das regiões, 2) visualização de dados da cidade (deliveries e hubs), 3) posicionamento de locais candidatos e 4) visualização dos resultados.

A Figura 9 apresenta a tela para selecionar as regiões. As regiões disponíveis são indicadas através de uma bola laranja. Para selecionar uma região, basta o clicar sobre a bola com o mouse. Após selecionar a região, a aplicação exibe um mapa da cidade com a localização das deliveries e dos hubs presentes na região (Figura 10).

A Figura 10) apresenta a tela com as informações (hubs e deliveries) de uma região e um formulário para realizar a consulta. A aplicação agrupa automaticamente os deliveries para facilitar a visualização. Por exemplo, uma bola cinza com o número 1.724 indica que há 1.724 devileries agrupados naquela região. Os ícones azuis com os coelhos exibem os locais dos hubs existentes atualmente.

A Figura 10) também contem um formulário para inserir os parâmetros da consulta. Ao habilitar o modo de edição, é possível clicar no mapa para posicionar os hubs candidatos. Este formulário também permite selecionar o algoritmo de localização ideal que será utilizado. Após posicionar os hubs candidatos e o algoritmo, deve-se clicar no botão "Start" para iniciar o processamento da consulta.

A Figura 11) exibe o resultado a consulta. Os ícones laranja no mapa são os locais onde os hubs candidatos foram posicionados. Uma bolinha vermelha com um número sobre o ícone do local candidato indica em que posição ele ficou. Em outras palavras, o melhor local candidato vai ter uma bolinha vermelha com o número 1. contem um formulário para inserir os parâmetros da consulta. Ao habilitar o modo de edição, é possível clicar no mapa para posicionar os hubs candidatos. Para cada um dos candidatos é exibido um resultado (e.x. soma das distâncias para o MinSum) e o número de deliveries que foram “atraídos” (que são mais próximos do candidato do que de qualquer outro hub existente) pelo candidato. Para iniciar uma nova consulta, basta clicar no botão "Resetar Pesquisa".

5 Considerações finais

Este artigo representa os resultados obtidos pelo projeto Hub point no ano de 2021 (maio a novembro). O primeiro desafio foi lidar com a grande quantidade de deliveries, o que torna o processamento dos algoritmos complexo. Para lidar com este problema, nós adaptamos os algoritmos para armazenar resultados parciais, reduzindo o tempo de processamento.

A aplicação também se mostrou essencial para que fosse possível visualizar os dados da consulta. Após essa primeira etapa, nós iremos desenvolver algoritmos que permitam utilizar uma função de distância mais precisa, como o caminho mínimo entre hubs e deliveries, o que vai proporcionar obter resultados mais precisos.

References

Xiaokui Xiao Bin Yao, Feifei Li. 2011. Optimal location queries in road network databases. *International Conference on Data Engineering (ICDE)* (2011).

Mônica Maria Mendes Luna Carolina Luisa dos Santos Vieira. 2016. Models and methods for logistics hub location: a review towards transportation networks design. *Archives of Transport* (2016).

Tian Xia Donghui Zhang, Yang Du. 2006. Progressive computation of the min-dist optimal-location query. *International Conference on Very Large Data Bases (VLDB)* (2006).

Stanisław Krzyżaniak, Marcin Hajdul, and Ireneusz Fechner. 2012. The Concept of a Logistics Centre Model as a Nodal Point of a Transport and Logistics Network. *Archives of Transport* (2012).

Venkata M.V. Gunturi Sarnath Ramnath. 2020. Optimal Load Balanced Demand Distribution under Overload Penalties. *arXiv* (2020).

Cihan Uyanık Senay Oguztimur, Gülfem Tuzkaya. 2018. A Literature Survey On Logistics Centers' Location Selection Problem. *Sigma Journal of Engineering and Natural Science* (2018).

Raymond Chi-Wing Wong Jiamin Xiong Ganglin Mai Cheng Long Zitong Chen, Yubao Liu. 2015. Optimal Location Queries in Road Networks. *ACM Trans* (2015).

Figure 9. Tela para seleção das regiões

Figure 10. Tela que exibe as informações de uma região (deliveries e hubs) e um formulário para realizar a consulta

Figure 11. Tela que exibe o resultado da consulta

Habilitando mineração de processos no domínio de operações logísticas

Extração de dados para um log de eventos centrado em objetos

Mateus Alex dos Santos Luna, Elio Ribeiro Faria Junior, Thais Rodrigues Neubauer
Marcelo Fantinato, Sarajane Marques Peres
{theew,elioribeirofaria,thais.neubauer,m.fantinato,sarajane}@usp.br
Escola de Artes, Ciências e Humanidades, Universidade de São Paulo
São Paulo, SP, Brasil

Figura 1: Engenharia de dados para mineração de processos no domínio de operações logísticas

RESUMO

Atualmente, pesquisadores em mineração de processos têm se preocupado em ir além da análise de processos isolados, uma vez que a complexidade dos processos organizacionais exige análise de múltiplos processos inter-relacionados. Mineração de processos centrada em objetos traz uma proposta conceitual para lidar com esse tipo de situação. Contudo, trata-se de um assunto do estado da arte na área, ainda com poucos exemplos de aplicação em contextos reais. Esse artigo relata um trabalho em andamento para aproximação do estado da prática ao estado da arte no contexto de mineração de processos centrada em objetos. A proposta é elaborar e instanciar um *framework* de engenharia de dados para construção de logs de eventos centrados em objetos no contexto prático de organizações que lidam com processos logísticos. Assim, neste artigo, nós apresentamos os primeiros resultados representados pelo modelo conceitual de dados orientado a um processo logísitico, pela instânciação desse modelo, e pela modelagem e instanciação de uma relação de mudanças cujo conteúdo reflete a operação de um sistema de informação sobre o modelo conceitual apresentado. A fase final de execução do trabalho aqui relatado inclui a proposição de procedimentos para uso da relação de mudanças para construção de um log de eventos centrado em objetos.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
PBP-LOGGI 2021, 8 de dezembro, 2021, São Paulo, SP
© 2021 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

CCS CONCEPTS

• **Information systems** → **Information lifecycle management**; • **Software and its engineering** → **Application specific development environments**; • **General and reference** → Design.

KEYWORDS

process mining, data engineering, object-centric event log

ACM Reference Format:
Mateus Alex dos Santos Luna, Elio Ribeiro Faria Junior, Thais Rodrigues Neubauer, Marcelo Fantinato, Sarajane Marques Peres. 2021. Habilitando mineração de processos no domínio de operações logísticas: Extração de dados para um log de eventos centrado em objetos. In *Workshop Virtual Loggi: Resultados do Programa de Bolsas de Pesquisa (PBP-LOGGI), 8 de dezembro, 2021, São Paulo, SP*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUÇÃO

Modelos de processos são ferramentas essenciais para o alcance de sucesso em gestão de processos em todos os tipos de organizações. Contudo, por razões culturais ou por falta de recursos humanos e materiais, é comum que os modelos de processos não sejam formalizados. Há situações em que os gestores dos processos não estão conscientes do fluxo real de atividades executadas no dia a dia das entidades que gerenciam. Nesse contexto, a análise automática de processos tem um papel crucial, pois fornece meios para que modelos de processos organizacionais sejam automaticamente descobertos, verificados e melhorados. Descoberta de modelos de processos e análises com objetivos de checagem de conformidade e de proposição de melhorias são tarefas estudadas na área de mineração de processos [19]. Em mineração de processos, a ciência de processos e a ciência de dados estabelecem um compromisso que permite o desenvolvimento de ferramental para apoio à gestão de processos organizacionais e tomadas de decisão associadas. Entretanto, a complexidade inerente aos processos organizacionais impede que a automação dessa tarefa alcance pleno sucesso. Duas

questões podem ser colocadas como fatores importantes na definição de ações em mineração de processos: o domínio do negócio e a natureza dos dados que fornecem as informações básicas para a mineração de processos dentro desse domínio de negócio.

Definir os métodos, as estratégias e as técnicas de mineração de processos adequados para cada domínio é um trabalho *ad hoc*. Não há um procedimento padronizado para isso, e muito do sucesso de iniciativas na área dependem do olhar que é dado ao problema de mineração dentro do negócio em questão. Especificamente na área de logística, há a expectativa de que exista alta complexidade, devido a pelo menos dois fatores: (a) processos diferentes que se inter-relacionam e (b) escalabilidade. Esse artigo tem como ponto de atenção o trabalho de engenharia de dados para suporte à mineração de processos inter-relacionados. Assim, aqui são apresentados os primeiros esforços no sentido de explorar modelagens de dados relacionais comumente usadas em softwares que suportam a execução de processos logísticos, e atrelar essa exploração a um levantamento de requisitos para construção de logs de eventos capazes de representar a dinâmica desses processos. Nessa primeira fase, o ponto de atenção está no primeiro fator citado.

Embora o conceito de mineração de múltiplos processos esteja em foco no estado da arte, ainda não faz parte do estado da prática em mineração de processos. Em geral, as ferramentas e consultorias adaptam as técnicas clássicas de mineração de processos para atender as especificidades de organizações. Organizações que trabalham com logística trazem um cenário adequado para o estudo da aplicação prática desses conceitos e esse projeto tem a intenção de explorar tal cenário. O objetivo da pesquisa é propor uma abordagem para o tratamento de dados no domínio da logística, que apoie a engenharia de dados referente à construção de logs de eventos adequados a esse tipo de negócio. Como primeira fase da pesquisa, é proposto o desenvolvimento de um *framework* para associar a construção de logs de eventos tradicionais e logs de eventos centrados em objetos à exploração de bases de dados relacionais comumente incorporadas a softwares de gestão organizacional.

Neste artigo, um modelo de dados conceitual, sua instanciação e associação com uma relação de mudanças (ou uma relação de auditoria) são apresentados no contexto de um sistema de gerenciamento de entregas fictício. Esses artefatos representam a base para o desenvolvimento de procedimentos de engenharia de dados que serão propostos para a construção de logs de eventos aderentes às necessidades da mineração de processos para contextos nos quais diferentes processos organizacionais se inter-relacionam.

2 REFERENCIAL TEÓRICO

Esta seção apresenta os conceitos básicos necessários para compreensão da proposição discutida neste artigo.

2.1 Mineração de processos - conceitos básicos

De acordo com [21], um processo de negócio pode ser definido como um conjunto de tarefas (ou atividades) executadas de forma coordenada dentro de ambientes organizacionais ou técnicos, com o objetivo de alcançar uma meta de negócio. Processos de negócio são submetidos a procedimentos de análise, monitoramento e melhoria considerando expectativas do gerenciamento de processos de negócio e seus ciclo de vida. Quando efetivo, o gerenciamento de processos de negócios garante resultados consistentes e agrega valor à organização e seus clientes [4].

A execução do ciclo de vida da gestão de um processo de negócio é primariamente feita com base em conhecimento tácito de gestores e especialistas ligados à organização, com vistas a procedimentos e regulações internas e externas à organização. Atualmente, esse contexto pode ser enriquecido se algum tipo de automação é adotado no suporte das tarefas do ciclo de vida do gerenciamento, considerando dados factuais e históricos sobre a execução dos processos. Tais dados estão registrados em log de eventos e são a base de uma disciplina denominada mineração de processos. A meta da mineração de processos de negócio é melhorar a compreensão sobre tais processos por meio da extração de conhecimento dos logs de eventos [18]. Logs de eventos são arquivos sequenciais nos quais eventos relacionados à execução das tarefas de um processo de negócio são registradas.

De acordo com [19], um evento $e \in \mathcal{E}$, em que $\mathcal{E}$ é o universo de eventos, corresponde à execução de uma tarefa $\tau \in \mathcal{T}$, em que $\mathcal{T}$ é o conjunto de atividades que ocorrem em um determinado processo de negócio. Eventos são comumente associados com atributos não-mandatórios (por exemplo, atividade, selo de tempo, recurso, custo e tipo de transação). Tipos de transação se referem à tarefas do ciclo de vida associado a um evento, e podem assumir valores como *START* and *COMPLETE*.

Um caso $c \in C$, em que C é o universo de casos, engloba uma sequência de eventos e^c. Um caso de refere a uma instância de processo que é uma execução real do processo. Todos os casos possuem um atributo mandatório chamado trace (σ). Um trace é uma sequência finita de eventos $\sigma \in \mathcal{E}^*$ e $1 \leq i < j \leq |\sigma| : \sigma(i) \neq \sigma(j)$. Um log de eventos L é um conjunto de casos $L \subset C$ e para qualquer $c_1, c_2 \in L$, se $c_1 \neq c_2$ então $\delta(e^{c_1}) \cap \delta(e^{c_2}) = \emptyset$, em que δ é um operador para converter uma sequência em um multi-conjunto. Um classificador é uma função que mapeia os atributos de um evento em um rótulo ou valor. Por exemplo, se $e.[attribute_notation]$ é adotado para indicar uma função classificadora, tem-se que $e.\tau$ indica o nome da tarefa associada a um evento e.

Mineração de processos usa logs de eventos como entrada para algoritmos que descobrem modelos de processos (descoberta), contrastam dados do log de eventos com informações de modelos de processo (checagem de conformidade) e extraem informações que permitem realizar melhorias no processo (melhoria de processos) [19]. Nesse artigo, estamos interessados em suportar a criação de logs de eventos centrados em objetos com o intuito de gerar informação referente a processos inter-relacionados para motivar a elaboração de estratégias de descoberta, checagem de conformidade e melhoria de processos inter-relacionados.

2.2 Mineração de processos inter-relacionados

Um dos primeiros trabalhos a ilustrar o problema de mineração de processos inter-relacionados foi apresentado em [6]. No seu estudo, os autores ilustraram a complexidade de processos inter-relacionados de vendas, entregas e cobranças usando conceitos de modelos de processos centrados em artefatos e referentes a Proclets [15]. Artefatos, nesse contexto, dizem respeito aos modelos de processos independentes criados para pontos de atenção específicos, por exemplo: o modelo de processo de vendas constitui um artefato,

o modelo de processo de entregas é um segundo artefato, e o modelo de processo de cobranças é um terceiro artefato. A lógica de execução de cada processo pode ser apresentada usando três redes de Petri e dependências comportamentais entre os processos podem ser modeladas usando portas e canais assíncronos provenientes da teoria de Proclets. Como uma alternativa, [14] aplicaram a teoria de *Guard-Stage-Milestones* [11]. Nessa iniciativa, o relacionamento entre as lógicas de execução dos processos de vendas, entregas e cobranças foi modelado usando uma linguagem declarativa capaz de expressar condições de verificação e de parada que controlam a dinâmica entrelaçada dos diferentes processos.

Uma estratégia simples e mais aderente aos conceitos básicos usados em mineração de processos é apresentada em [13] para permitir fluxos de controle entre processos independentes. Essa estratégia considera dependências comportamentais e a frequência com que elas ocorrem. A estratégia semiautomática apresentada nesse trabalho inicia na identificação de artefatos (pontos de atenção) e da fonte de dados (p.ex. modelos relacionais) e finaliza na apresentação do modelo de processo AS-IS. Em [5], o conceito de modelos de processos centrados em artefatos é usado junto a estratégias de descoberta de modelos de processo expressos em máquinas de estados. Essa pesquisa usa log de eventos que integram os eventos dos processos referentes a cada artefato e permitem descobrir um modelo de processo único. A origem dos eventos é preservada no modelo de forma que é possível explorar as interações entre os múltiplos artefatos.

Também em 2017, o conceito de mineração de dados orientado a objetos é apresentado em [17]. Esse conceito representou uma evolução no trabalho com múltiplos processos (agora denominados objetos), pois incorpora modelagem de dados relacionada ao log de eventos, o que permite trabalhar com relações de múltiplas cardinalidades ($m : n$) e implementar algoritmos de checagem de conformidade e análise de desempenho. Esse conceito sofreu evoluções de forma que modelos de múltiplos pontos de vistas puderam ser desenvolvidos [2], culminando em 2020 na apresentação das redes de Petri centradas em objetos [16]. Essas redes de Petri são redes coloridas nas quais é possível visualizar todas as relações existentes entre elementos de diferentes processos.

A evolução dos estudos em mineração de processos gerou uma série de padrões para logs de eventos, cada qual apresentando, no decorrer do tempo, características para acomodar novos conceitos de representação de informação. A partir da identificação da necessidade de trabalhar com processos inter-relacionados, os padrões para log de eventos passaram a incluir características especializadas, dando origem a uma família de padrões que, atualmente, estão no centro das atenções da engenharia de dados da mineração de processos. A adoção desses novos padrões ainda não é unanimidade entre pesquisadores e práticos da área.

2.3 Padrões para log de eventos

A figura 2 resume a evolução dos padrões de log de eventos. O primeiro padrão a ser estabelecido foi o MXML, um padrão no qual as informações dizem respeito a um único processo. O último padrão, o OCEL, permite organizar informações sobre múltiplos processos relacionados. A figura ainda posiciona nessa linha do tempo, conceito e padrões de propósito geral que foram usados pela comunidade de mineração de processos, ou como base ou como uma alternativa aos padrões específicos da área. Os padrões especificamente criados para log de eventos seguem descritos no restante desta seção.

2.3.1 O padrão MXML: Mining eXtensible Markup Language. Mineração de processos é uma disciplina relativamente nova que teve seus primeiros conceitos e métodos estabelecidos no início dos anos 2000, e que rapidamente se desenvolveu dentro das comunidades acadêmica e, posteriormente, industrial. Devido a esse crescimento, a área de mineração de processos precisou estabelecer um metamodelo para padronizar a organização da informação referente ao log de eventos [3]. O primeiro padrão para log de eventos estabelecido foi o MXML. Um grande problema encontrado no padrão MXML foi a dificuldade em lidar com a padronização de atributos adicionais àqueles já previstos em sua sintaxe [3, 9]. Uma versão semanticamente anotada do MXML foi então desenvolvida, o SA-MXML (*Semantically Annotated Mining eXtensible Markup Language*), porém o nível de complexidade inerente ao padrão impediu que ele fosse amplamente adotado.

2.3.2 O padrão XES: eXtensible Event Stream. Para resolver as limitações do MXML, um novo padrão foi desenvolvido [12]: o XES. Nesse padrão, um atributo é um par chave-valor, cujo tipo pode ser um texto, um número inteiro ou real, uma data ou hora, um tipo lógico, um identificador ou uma lista de atributos. Para fornecer semântica aos dados, o XES permite a criação de extensões. Tais extensões definem chaves e fornecem um significado para os valores associados a essas chaves. Segundo o exposto em [10], o padrão XES foi estabelecido para atingir quatro objetivos:

- Simplicidade: representar as informações da forma mais simples, com o propósito de facilitar o entendimento por parte do usuário.
- Flexibilidade: poder ser utilizado em qualquer domínio de aplicação além da mineração de processos.
- Extensibilidade: realizar atualizações no padrão sem perder a compatibilidade com versões anteriores.
- Expressividade: empregar um formato genérico no padrão, mas com o mínimo de perda de especificidade de informação.

A figura 3 ilustra o metamodelo para o padrão XES. Nele, o log de eventos é composto por um ou mais traços, associados a casos, que descrevem a ordem de execução das atividades relacionadas ao processo. Um traço pode conter eventos, os quais representam as atividades do processo. Essa estrutura de log de eventos, traço e evento possuem atributos que são utilizados para armazenar as informações, por exemplo, um evento pode conter um atributo do tipo *string* responsável por armazenar o nome da atividade associada ao evento. Há também a composição de um classificador de atributos, pois os atributos não são predefinidos. Por fim, o log de eventos contém uma extensão para determinar o conjunto de atributos que será utilizado em um domínio de aplicação específico [10].

2.3.3 O padrão XOC: eXtensible Object-Centric. Logs de eventos XES organizam os eventos com uma noção de caso, ou seja, o log de eventos é orientado a um processo. Considerando que em muitos cenários reais há a necessidade de analisar processos interrelacionados, diferentes noções de casos passam a ser importantes. O padrão XES, ao assumir apenas uma única noção de caso, exige

Figura 2: Evolução de padrões para log de eventos

Figura 3: Metamodelo do padrão XES [10]

que a informação seja "planificada", gerando os problemas conhecidos como deficiência, convergência e divergência de informação no log de eventos [20]. O padrão XOC foi criado para resolver o problema de uso de múltiplas noções de casos. Nesse padrão, eventos são descritos com um identificador próprio (nome da atividade associada a ele), com uma lista de objetos ao qual o evento se refere e uma estrutura que representa a relação entre objetos no momento em que o evento ocorre.

2.3.4 O padrão OCEL: Object Centric Event Log. O log de eventos centrado em objeto organizado no padrão XOC pode gerar problemas de desempenho [8]. A fim de superar essa dificuldade, o padrão OCEL foi estabelecido [7]. O padrão OCEL suporta informação sobre eventos relacionados a múltiplos processos e permite que a inter-relação entre os processos seja observada. A tabela 1 mostra representações informais da informação presente em um log de eventos organizado no padrão OCEL. A primeira parte da tabela representa os eventos, por exemplo, o *evento 1* está associado a uma *atividade*, ao *dia e horário* da sua realização e aos *objetos* do tipo: **pedido**, **item** e **cliente**. A segunda parte da tabela mostra os objetos. Nela, o valor *i1* é um objeto do tipo **item** e possui os atributos **produto** e **peso**; isso significa que o *evento 1* é referente a *atividade* "realizar pedido", realizada em "10-09-2021 às 08:05:03", pelo colaborador "Pedro", e essa atividade faz referências aos objetos *p1*, *i1*, *i2* e *c1*.

Tabela 1: Representações no padrão OCEL, adaptado de [8]

Eventos	Atividades	*Timestamp*	Pedido	Item	Cliente
evento 1	Realizar pedido	10-09-2021 08:05:03	p1	i1, i2	c1
evento 2	Confirmar pedido	10-09-2021 08:15:55	p1	-	-
evento 3	Verificar disponibilidade	10-09-2021 08:30:01	p1	i1	-
Objetos	Tipo	Produto	Peso	Idade	ID Conta
i1	item	TV	18,00	-	-
i2	item	Geladeira	37,00	-	-
c1	cliente	-	-	30	9874

A figura 4 mostra o metamodelo do padrão OCEL. Nesse padrão, o log de eventos é composto por eventos e objetos. Eventos possuem um ou mais objetos e o log de eventos pode assumir atributos globais. Os eventos e objetos são compostos por elementos que podem ser do tipo texto, data e hora, número inteiro ou real e do tipo lógico. Para suportar o trabalho com o padrão OCEL está disponível uma ferramenta[1] e um plugin para a ferramenta ProM[2]. Esses recursos são capazes de importar e exportar nas serializações JSON e XML, o conteúdo de um log de eventos organizado no padrão OCEL [7].

3 MÉTODO

O *framework* de engenharia de dados proposto nesse trabalho pressupõe a criação de um contexto sintético para execução da extração de informação para os logs de eventos. Para a criação do contexto sintético, um sistema simplificado referente ao contexto de operações logísticas foi projetado. Para isso, três artefatos foram criados:

[1]https://github.com/OCEL-standard/ocel-support
[2]Pacote: OCEL Standard (ProM 6.10).

Figura 4: Metamodelo do padrão OCEL (adaptado de [7])

- um modelo de dados conceitual para armazenamento de informações referentes à execução de operações logísticas;
- um projeto de inserção de dados orientado à diferentes situações possíveis na execução das atividades referentes às operações logísticas;
- construção e instanciação de um modelo físico de dados para geração da instância de banco de dados que fornecerá os insumos para criação de logs de eventos.

Uma vez estabelecido esse contexto sintético, os próximos passos para conclusão desse trabalho são:

- modelar logs de eventos seguindo a especificação do padrão XES, a especificação do padrão XOC e a especificação do padrão OCEL;
- construir *scripts* de extração de informação do modelo conceitual e da relação de mudanças a fim de instânciar os logs de eventos;
- evidenciar as diferenças entre os logs de eventos construídos sob à perspectiva de tarefas de descoberta de modelos de processos, checagem de conformidade e execução de melhoria de processos.

4 RESULTADOS

A seção 4.1 apresenta o contexto sintético referente ao domínio logístico suportado por um sistema de informação simplificado e o respectivo modelo de dados conceitual para armazenamento de dados associados às atividades executadas no referido domínio. A seção 4.2 descreve variações de instanciação do modelo de dados, simulando algumas situações reais que podem ocorrer no domínio de aplicação modelado. A seção 4.3 mostra um excerto da relação de mudanças que representa a entidade fornecedora de informação para a construção de um log de eventos[3].

4.1 Modelo de dados: operações logísticas

O modelo de dados utilizado foi criado para refletir uma situação relacionada ao contexto logístico, porém de forma simplificada. Seguindo essa ideia, foi definido o modelo ilustrado na figura 5.

[3]Banco de dados disponível em: https://github.com/pm-usp/data-engineering

O modelo representa uma organização que recebe pedidos de entregas de diferentes itens, possivelmente atrelados a um único pedido no lojista. Porém, na visão simplificada do sistema projetado, os pedidos estão associados ao lojista e têm atributos indicando a origem do pedido, e o destino e destinatário dos items. Cada item, de forma individual, pode ser colocado em um pacote ou podem ser empacotados em conjunto. Os pacotes são as unidades entregues aos destinatários. Para registrar as diferentes tentativas de entregas, possivelmente realizadas por diferentes entregadores, há uma relação que registra as entregas e o estado dela. Os lojistas enviam itens para entrega, e a organização que utliza o sistema tem diferentes motivos para desejar ter informação sobre o lojista: desde priorização de lojistas parceiros até redução da redundância no modelo. Os entregadores, que levam os pacotes até seus destinatários, precisam também estar presentes no modelo de dados para que a orgnaização logística possa controlar quais entregas cada um deles realizou.

4.2 Variantes dentro do modelo de dados

A instanciação do modelo de dados conceitual considera situações variadas. Por exemplo, como um pacote não está restrito a um item, itens de diferentes pedidos podem ser empacotados juntos caso tenham um mesmo destinatário e itens que pertencem a um mesmo pedido podem ser entregues em pacotes separados. As entregas podem falhar, quando, por exemplo, não há ninguém para receber o pedido e, assim, um mesmo pacote pode constar em diferentes tentativas de entregas até que uma delas seja bem-sucedida. Também há a possibilidade de uma entrega ter vários pacotes, como quando uma primeira entrega de um pacote falha e outro chega para ser entregue ao mesmo destinatário. Para este trabalho, tentamos cobrir algumas dessas variações para identificar como elas influenciam a extração de dados para construção dos logs de eventos.

4.3 A relação de mudanças

Definidas as relações do sistema de informação, uma relação adicional foi criada para registrar as interações do sistema com o banco de dados: a relação nomeada "*Change table*". Essa relação é composta por atributos que registram o tipo de alteração realizado nas relações do banco de dados, a relação afetada pela alteração, os valores alterados e quando essas alterações foram efetuadas. Um exemplo de registro gravado na relação "*Change table*" é mostrado na figura 6. O registro nessa relação é criado a partir de um gatilho que identifica alterações em todas as outras relações do banco de dados. Como ação do gatilho são inseridas informações sobre as mudanças nas relações do sistema de informação. Esse tipo de relação também é conhecida como relação de auditoria. Podemos ver na figura um exemplo da inclusão de uma entrega ocorrendo nas duas primeiras linhas: é criada uma entrega e um pacote é relacionado a ela, e ambas as ações são registradas na "*Change table*". Em seguida, uma outra entrega recebe uma atualização de seu *status*: ela falha, e em seguida uma outra entrega é criada para entregar os pacotes que a anterior falhou em entregar, além de possíveis novos pacotes que vão para o mesmo destinatário.

5 CONSIDERAÇÕES FINAIS

Esse artigo apresentou os progressos realizados em um projeto de engenharia de dados para mineração de processos. Até o momento,

Figura 5: Modelo de dados conceitual - operações logísticas

	act character varying (10)	tabela character varying (40)	valor_anterior character varying (120)	valor_atual character varying (120)	campo character varying (40)	timestamp timestamp with time zone
28	INSERT	entrega	[null]	("A entregar",2021-11-26,3,3)	all	2021-11-26 09:16:13.631207-03
29	INSERT	pacote_entrega	[null]	(1,3)	all	2021-11-26 09:16:13.631207-03
30	UPDATE	entrega	"A entregar"	"Falha"	status	2021-11-26 09:29:04.979601-03
31	INSERT	entrega	[null]	("A entregar",2021-11-27,2,2)	all	2021-11-26 09:38:49.759988-03
32	INSERT	pacote_entrega	[null]	(2,2)	all	2021-11-26 09:38:49.759988-03
33	INSERT	pacote_entrega	[null]	(3,2)	all	2021-11-26 09:38:49.759988-03

Figura 6: Exemplo de registros na relação de mudanças

as ações referentes ao projeto permitiram fazer um levantamento das principais características de diferentes padrões de logs de eventos usados em mineração de processos e projetar um contexto sintético para suportar a proposição de um *framework* para extração de dados e construção de logs de eventos. Nos próximos passos, diferentes padrões serão usados para construir logs de eventos a partir dos dados presentes na instância do contexto sintético de operações logísticas. Este trabalho está especialmente interessado na discussão da utilidade de logs de eventos centrados em objetos e em fornecer subsídios para o desenvolvimento de mineração de processos aderente ao contexto de operações logísticas. Um exemplo de extração de logs de eventos baseado em sistemas corporativos atuais foi desenvolvido em [1]. Esse exemplo será a base para os próximos passos deste trabalho.

6 AGRADECIMENTOS

Este trabalho foi realizado com o apoio da Loggi (www.loggi.com), por meio do Programa de Bolsas de Pesquisa Loggi (PBP-LOGGI).

REFERÊNCIAS

[1] Alessandro Berti, Gyunam Park, Majid Rafiei, and Wil van der Aalst. 2021. An Event Data Extraction Approach from SAP ERP for Process Mining. arXiv:2110.03467 [cs.DB]
[2] A. Berti and W. van der Aalst. 2018. StarStar Models: Process Analysis on top of Databases. In *In CEUR Workshop*.
[3] B. V. Dongen and W. Aalst. 2005. A Meta Model for Process Mining Data. In *Open Interop Workshop on Enterprise Modelling and Ontologies for Interoperability, Co-located with CAiSE'05 Conference*.
[4] Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A. Reijers. 2013. *Fundamentals of Business Process Management.* Springer Publishing Company, Incorporated.
[5] Maikel Eck, Natalia Sidorova, and Wil var der Aalst. 2019. Guided Interaction Exploration and Performance Analysis in Artifact-Centric Process Models. *Process Models. Bus. Inf. Syst. Eng.* 61 (2019), 649––663.
[6] D. Fahland, M. de Leoni, B. F. van Dongen, and W. van der Aalst. 2011. Behavioral Conformance of Artifact-Centric Process Models. In *Lecture Notes in Bus. Inf. Process.* Springer, 37–49.
[7] A. F. Ghahfarokhi, G. Park, A. Berti, and W. van der Aalst. 2020. Ocel standard.
[8] Anahita Farhang Ghahfarokhi, Gyunam Park, Alessandro Berti, and Wil M. P. van der Aalst. 2021. OCEL: A Standard for Object-Centric Event Logs. In *New Trends in Database and Information Systems.* Springer, 169–175.
[9] C. W. Günther. 2009. *Process mining in flexible environments.* Ph. D. Dissertation. Technische Universiteit Eindhoven.
[10] C. W. Gunther and H. M. W. Verbeek. 2014. *XES - standard definition.* BPMcenter.org.
[11] R. Hull, E. Damaggio, R. De Masellis, F. Fournier, M. Gupta, F. T. Heath, S. Hobson, M. Linehan, S. Maradugu, A. Nigam, P. N. Sukaviriya, and R. Vaculin. 2011. Business artifacts with guard-stage-milestone lifecycles: Managing artifact interactions with conditions and events. In *Proc. of Intl. Conf. on Distributed Event-Based Syst.* 51–62.
[12] IEEE. 2016. IEEE Standard for eXtensible Event Stream (XES) for Achieving Interoperability in Event Logs and Event Streams - IEEE Std 1849-2016. , 50 pages.
[13] X. Lu, M. Nagelkerke, D. van de Wiel, and D. Fahland. 2015. Discovering Interacting Artifacts from ERP Systems. *IEEE Trans. on Serv. Comp.* (2015), 861–873.
[14] Viara Popova and Marlon Dumas. 2014. Discovering Unbounded Synchronization Conditions in Artifact-Centric Process Models. In *Bus. Process Manage. Workshops. Lecture Notes in Bus. Inf. Process.* Springer.
[15] W. van der Aalst, P. Barthelmess, C. A. Ellis, and J. Wainer. 2001. Proclets: A Framework for Lightweight Interacting Workflow Processes. *Int. J. of Cooperative Inf. Sys.* 10 (2001), 443–481.
[16] W. van der Aalst and A. Berti. 2020. Discovering Object-Centric Petri Nets. *Fundamenta Informaticae* XXI (2020), 1001–1040.
[17] W. van der Aalst, G. Li, and M. Montali. 2017. Object-Centric Behavioral Constraints. In *CEUR Workshop Proc. in Description Logics.*
[18] W. M. P. van der Aalst. 2011. *Process mining – discovery, conformance and enhancement of business processes* (1st ed.). Springer, Berlin, Germany.
[19] Wil M. P. van der Aalst. 2016. *Process Mining - Data Science in Action.* Springer.
[20] Wil M. P. van der Aalst. 2019. *Object-Centric Process Mining: Dealing with Divergence and Convergence in Event Data.* 3–25. https://doi.org/10.1007/978-3-030-30446-1_1
[21] M. Weske. 2007. *Business Process Management: Concepts, Languages, Architectures* (2nd ed.). Springer.

Alto nível vs baixo nível: Uma abordagem C++ para o problema de otimização da plataforma loggibud

FELLIPE PESSANHA*, Universidade Federal Fluminense, BR
IGOR MACHADO COELHO, Universidade Federal Fluminense, BR
LUIZ SATORU OCHI, Universidade Federal Fluminense, BR

Este trabalho aborda aspectos práticos de implementação, manutenção e projeto de sistemas de otimização, com foco em roteamento de veículos. O problema clássico de roteamento de veículos (PRV), por ser NP-Difícil, enseja abordagens com baixo tempo computacional para uso prático, por isso abordamos heurísticas. Em especial, abordamos a possibilidade prática de aplicação de técnicas heurísticas inovadoras, com o Multi Improvement, desenvolvida por pesquisadores brasileiros nos últimos anos. Devido à complexidade do método e do problema abordado, também abordamos o uso do arcabouços de otimização OptFrame, que facilita a implementação e testes de heurísticas na linguagem C++. Resultados indicam a viabilidade prática (em baixo tempo computacional) de uso da técnica Multi Improvement como alternativa a estratégias clássicas de exploração por busca local, bem como abre portas para extensões futuras que utilizem programação paralela e outras heurísticas de otimização. Dada a grande escala do problema, propomos a variante aleatória da descida com Multi Improvement, denominada Multi Random Selection, e comparamos com a descida aleatória clássica, chamada Random Descent Method. O estudo também aborda as especificidades de utilizar linguagens de baixo nível para implementar metaheurísticas: a capacidade de análise proporcionada por elas, as dificuldades intrínsecas em programar com esse tipo de ferramenta, e os possíveis ganhos de performance ao utilizá-las

Additional Key Words and Phrases: Multi Improvement, Problema de Roteamento de Veículos, Arcabouços de Otimização

ACM Reference Format:
Fellipe Pessanha, Igor Machado Coelho, and Luiz Satoru Ochi. 2018. Alto nível vs baixo nível: Uma abordagem C++ para o problema de otimização da plataforma loggibud. *ACM Trans. Graph.* 37, 4, Article 111 (August 2018), 6 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUÇÃO

O projeto "Estratégias Inovadoras de Exploração por Multi Improvement para Roteamento de Veículos da Loggi" aborda possibilidades de técnicas de otimização por meio de heurísticas, em especial usando a técnica recentemente proposta de Multi Improvement, no contexto da da plataforma de benchmark *loggibud*, mantido pela empresa Loggi.

O problema clássico de roteamento de veículos [Dantzig and Ramser 1959] aborda uma extensão do problema do caixeiro viajante [Applegate et al. 2011], na qual capacidades são atribuídas a veículos que devem visitar um conjunto de clientes bem como satisfazer suas demandas, com tempo mínimo na rota. Este problema se desdobrou em diversas variantes ao longo dos anos, incluindo características diversas, como janelas de tempo e frotas heterogêneas [Penna 2013]. Ainda assim, variantes com rotas únicas se mostram ainda com grande dificuldade de resolução prática, especialmente quando um número elevado de clientes é considerado, levando em conta a natureza NP-Difícil do problema [Garey and Johnson 1979]. Nesse sentido, embora existam abordagens cada vez melhores para resolução exata desse tipo de problema, acaba sendo necessário o uso de estratégias heurísticas (sem garantia de otimalidade) quando o tempo computacional de resolução é um fator crítico (tipicamente para decisões operacionais de curto prazo, em contraste a decisões estratégias de longo prazo).

No trabalho buscamos não somente contribuir com proposição de algoritmos no contexto da Loggi, mas também buscando fazer uma implementação própria da resolução do problema através de heurísticas, utilizando a linguagem de programação C++, junto com framework de otimização OptFrame [Coelho et al. 2010, 2020]. A combinação deveria permitir um desenvolvimento rápido e eficiente de software de otimização. Como foco principal, consideramos avanços recentes em uma técnica de otimização desenvolvida por colaboradores deste trabalho, chamada de Multi Improvement [Rios et al. 2016]. Devido ao grande número de clientes considerado em problemas reais (em contraste com problemas de ordem mais teórica), apresentamos também uma abordagem de pré-processamento por clusterização, de forma a reduzir significativamente o volume de consultas ao sistema de mapas, responsável pela gestão da base de coordenadas dos clientes. Propomos assim uma variante do Multi Improvement com movimentos aleatórios, uma estratégia ainda inovadora na literatura, e que demonstra resultados promissores para exploração heurística e busca local.

Este trabalho é organizado em 4 seções, além desta introdução. A Seção 2 apresenta trabalhos relacionados. Na Seção **??** é apresentada a metodologia do trabalho, discutindo os diversos pontos que o desenvolvimento desse projeto precisou abordar, e incluindo a proposta do Multi Random Descent, um novo método explorativo para meta heurísticas, baseado no Multi Improvement. A Seção 4 mostra os resultados obtidos pelo software que já foi desenvolvido. Traz discussões acerca da direção para a qual os resultados preliminares

*Nota: Esse artigo foi produzido com atividades no período de 6 (seis) meses no projeto (Maio/2021 a Outubro/2021) acumulando assim os resultados de estudo bibliográfico e implementações no tema. O trabalho foi executado ao longo do tempo percorrido da bolsa. Além do apoio da Loggi, os autores agradecem à CAPES, FAPERJ e CNPq, por suportarem parcialmente a infraestrutura de pesquisa utilizada nessa pesquisa, bem como por bolsas de apoio individual aos pesquisadores envolvidos.

Authors' addresses: Fellipe Pessanha, fpessanha@id.uff.br, Universidade Federal Fluminense, Niterói, Rio de Janeiro, BR; Igor Machado Coelho, Universidade Federal Fluminense, Niterói, Rio de Janeiro, BR, imcoelho@ic.uff.br; Luiz Satoru Ochi, Universidade Federal Fluminense, Niterói, Rio de Janeiro, BR, satoru@ic.uff.br.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2018 Association for Computing Machinery.
0730-0301/2018/8-ART111 $15.00
https://doi.org/10.1145/1122445.1122456

apontam, mostrando a capacidade de crescimento do trabalho já feito. Finalmente, a Seção 5 trata dos próximos passos que devem ser tomados para o seguimento da pesquisa.

2 TRABALHOS RELACIONADOS

A área de otimização busca resolver problemas desafiadores dentro dos critérios especificados por um tomador de decisão. Tipicamente, problemas operacionais práticos exigem rápida tomada de decisão, motivando portanto o desenvolvimento de estratégias heurísticas genéricas e flexíveis, denominadas meta-heurísticas [Glover and Kochenberger 2006]. Outras estratégias de programação podem ser utilizadas para acelerar o processo de busca, explorando aspectos de baixo nível da arquitetura computacional disponível.

O uso de computação paralela, especialmente das *Graphics Processing Units* (GPUs), tem tido avanços com novas linguagens de programação, como CUDA [Sanders and Kandrot 2010]. Apesar dos avanços no contexto CPU-GPU, abordagens clássicas de busca heurística somente se utilizam das CPUs, com processamento sequencial (ou parcialmente paralelo, como multi-cores). Na realidade, o uso de processadores massivamente paralelos como GPUs tem muito a oferecer, como visto recentemente na literatura, porém existe que algoritmos sejam re-escritos e, muitas vezes, repensados do zero. Nesse espírito, algoritmos clássicos de otimização podem ser revistos no olhar de novas arquiteturas de computação, fora do "espírito sequencial", considerado como um padrão natural imutável.

O algoritmo Multi Improvement (MI) foi proposto em 2016 por [Rios et al. 2016], como uma maneira de explorar vizinhanças em problemas de otimização, através de métodos CPU-GPU. Devido à natureza de *Single Instruction Multiple Data* (SIMD) das GPUs, é necessário que os dados sejam massivamente processados simultaneamente, porém isso exige que um processo não interfira no outro. Também foi observado que, técnicas clássicas de exploração de vizinhança como First Improvement (FI) e Best Improvement (BI) consideram uma estratégia de busca por trajetória, onde apenas um único movimento de melhora é escolhido a cada passo. No BI, o melhor vizinho (ou melhor movimento) é escolhido, enquanto no FI, o primeiro vizinho de melhora (ou primeiro movimento de melhora) é adotado. O FI tende a se mover mais rapidamente que o BI, exigindo um maior número de iterações, porém com custo reduzido em cada iteração [Hansen and Mladenović 2006]. Por outro lado, o MI busca aplicar simultaneamente todos movimentos de melhora que não conflitem uns com os outros, obtendo ganhos significativos, a um custo maior por iteração. Dada uma função de avaliação f, temos que movimentos m_1 e m_2 são ditos *independentes* (ou não conflitantes, denotado por $m_1||m_2$) quando, para uma solução s qualquer: $\hat{m}_1 = f(m_1(s)) - f(s) = f(m_1(m_2(s))) - f(m_2(s))$ e $\hat{m}_2 = f(m_2(s)) - f(s) = f(m_2(m_1(s))) - f(m_1(s))$. Esta modelagem pode ser encontrada em detalhes na Tese [Rios 2016] e na Dissertação [Araújo 2018]. O Algoritmo 1 apresenta o Multi Improvement.

A estratégia mais comum do Multi Improvement, chamado de *Multi Improvement Heurístico*, consiste em escolher o movimento com maior ganho (estratégia gulosa) e, posteriormente, invalidar movimentos candidatos que sejam conflitantes com os movimentos previamente adotados. Essa abordagem foi recentemente demonstrada para um problema de roteamento de veículos [Júnior et al. 2019], com caráter ainda bastante experimental e poucos problemas-teste considerados. Recentemente, o Multi Improvement foi explorado no contexto de resolução exata através de programação dinâmica, sendo formulado um problema denominado *Maximum Multi Improvement Problem* [Nascimento Silva et al. 2020]. Essa estratégia tem um potencial ainda maior de obter grandes avanços na seleção de movimentos independentes, porém introduzindo um custo bastante significativo no custo de cada iteração do método. Naturalmente, no contexto de busca local, o MI pode ser introduzido juntamente com meta-heurísticas clássicas, como *Iterated Local Search* [Lourenço et al. 2003] e *Variable Neighborhood Search* [Mladenović and Hansen 1997].

Algorithm 1: MultiImprovement($f()$, $N()$, s)

```
/* Iterador de movimentos para vizinhança N                    */
m^it ← N(s);
/* Armazena em M pares de movimentos e custos                  */
M ← {};
foreach m ∈ m^it do
    /* Calcula custo m̂ do movimento m                          */
    m̂ ← f(m(s)) − f(s);
    /* Somente aceita movimentos de melhora                    */
    if (m̂ < 0) then
        M ← M ∪ {(m̂, m)};
/* Ordena M pelo primeiro valor dos pares                      */
M^Sorted ← OrdenaPrimeiro(M);
/* Multi Improvement guloso heurístico                         */
s* ← s;
for k ∈ {1, 2, ..., |M^Sorted|} do
    (m_k, m̂_k) ← M_k^Sorted;
    /* Aplica primeiro movimento disponível                    */
    s* ← m_k(s*);
    /* Atualiza movimentos independentes                       */
    for j ∈ {k + 1, ..., |M^Sorted|} do
        (m_j, m̂_j) ← M_j^Sorted;
        /* Elimina movimentos conflitantes                     */
        if (m_k ∦ m_j) then
            M^Sorted ← M^Sorted \ (m_j, m̂_j);
return s*;
```

3 METODOLOGIA

O contato com os dados utilizados na pesquisa foi feito através do repositório da loggibud no github[1]. Lá encontramos um programa de busca local escrito na linguagem interpretada e de alto nível Python, que oferece diversas facilidades em todo o processo de tratamento de dados em troca de uma queda considerável em desempenho.

[1]https://github.com/loggi/loggibud

O jeito que os desenvolvedores têm de lidar com essa "lentidão" é utilizando bibliotecas que encapsulam trabalhos mais complicados de performance em *bindings* para outras linguagens de mais baixo, como Fortran, C, e C++, tentando assim manter um bom equilíbrio de performance e velocidade de desenvolvimento. Porém, em situações onde a performance é crítica, como em problemas de otimização, ou onde há a exigência de uma customização maior, as linguagens de mais baixo nível se fazem necessárias, pois a customização do software fica extremamente comprometida quando se leva em consideração as soluções "fechadas" que os *bindings* representam. Por essa razão, adotamos soluções Python para problemas bem resolvidos em alto nível, ao mesmo tempo em que codificamos em C++ a estratégia de otimização heurística, através do arcabouço OptFrame.

3.1 Clustering

Uma dificuldade extra de atacar esse problema estava no grande volume de paradas, que podiam chegar a 32 mil em uma única instância. Esse volume impossibilitava, por exemplo, o armazenamento das distâncias em uma matriz quadrada de distância dois a dois, conectando todos os pontos da instância. Para contornar esse problema, decidimos usar um algoritmo de *clusterização*, que dividia a instância com N paradas em k subconjuntos disjuntos de paradas, cada parada i tendo n_i paradas. Dessa maneira, a memória necessária para armazenamento se tornava suportável pelas máquinas que temos disponíveis, já que:

$$N^2 = \left(\sum_{i \leq k} n_k\right)^2 = \sum_{i \leq k} {n_k}^2 + 2 \cdot \sum_{i,j \leq k} n_i \cdot n_j > \sum_{i \leq k} {n_i}^2$$

A clusterização foi feita utilizando a biblioteca *sklearn* em Python. Esse tipo de trabalho, que é feito poucas vezes, e já não leva muito tempo é o tipo de situação onde linguagens como o Python mostram o seu valor. Na Figura 1 podemos ver a representação gráfica da clusterização de uma instância. As distâncias utilizadas no processo de clusterização foram euclideanas, baseadas nas coordenadas do ponto. Novamente, a quantidade grande de paradas se mostrou um problema durante o ataque desse problema, ao impossibilitar a implementação da distância real calculada pelo *Open Source Routing Machine*[2] (OSRM) durante a clusterização com as máquinas disponíveis.

3.2 Adaptação do Multi Improvement

O escopo inicial do trabalho era o de utilizar de meta-heurísticas para resolver, com os dados da plataforma loggibud, problemas de roteamento de veículos, com o objetivo de implementar as heurísticas que têm, historicamente, os melhores resultados. Isso se soma à ideia de avaliar, através do Multi Improvement, a *independência de movimentos*, ou seja, quais dos movimentos gerados através de uma determinada vizinhança podem ser aplicados paralelamente em uma solução, tendo aproveitamento equivalente ao dos dois movimentos separados.

[2]http://project-osrm.org/

Fig. 1. Representação gráfica da clusterização da instância rj-0

Como proposto, a execução do multi improvement depende de um iterador que aplicasse todos os movimentos em uma determinada vizinhança, similar ao modelo de exploração por Best Improvement. Essa execução se mostrou pouco prática em problemas tão complexos como os aqui propostos, que geraram a necessidade de buscar uma alternativa para a inserção paralela de movimentos independentes.

A solução para isso se deu na criação do método *Multi Random Selection* (MIRS). O método consiste em aplicar diversos movimentos aleatórios em uma solução, encontrar dentre as soluções achadas na vizinhança, quais são as independentes que melhoram o *score* da solução atual, com limitação a um parâmetro de entrada *mvCount*. Tal parâmetro permite uma exploração mais profunda ou mais rasa, à medida que os problemas-teste se tornam mais complexos. O Algoritmo 2 descreve o método proposto MIRS.

3.3 Heurísticas Construtivas e Vizinhanças

Após conseguir dividir o problema em clusters, e armazenarmos as distâncias entre pontos dentro de um mesmo cluster em memória, conseguimos iniciar um processo de busca viável com as meta heurísticas. Iniciamos esse processo com um algoritmo construtivo baseado na heurística de Clarke Wright [Clarke and Wright 1964], aplicado para cada cluster, e já gerando soluções de qualidade relativamente altas.

Devido à grande dificuldade de implementação de vizinhanças interrotas, precisamos nos ater a movimentos intrarrotas no presente trabalho. Escolhemos utilizar apenas o movimento 2-Opt devido ao seu desempenho extremamente superior em relação às outras [Penna 2013].

3.4 Descidas, Descidas Aleatórias e Recozimento Simulado

Na literatura, o *Random Descent Method* [Souza 2008], conhecido como RDM, é um método clássico de descida na qual movimentos aleatórios são empregados, enquanto houver melhora da solução corrente. Ele é um método tipicamente empregado em problemas

Algorithm 2: MIRS($f()$, $N()$, $mvCount$, s)

```
/* Iterador de movimentos para vizinhança N */
m^it ← N(s);
/* Armazena em M pares de movimentos e custos */
M ← {};
i ← 0;
while i < mvCount do
    /* Gera movimento aleatório na vizinhança */
    m ← random(m^it);
    /* Calcula custo m̂ do movimento m */
    m̂ ← f(m(s)) − f(s);
    /* Somente aceita movimentos de melhora */
    if (m̂ < 0) then
        M ← M ∪ {(m̂, m)};
        i ← i + 1;
/* Ordena M pelo primeiro valor dos pares */
M^Sorted ← OrdenaPrimeiro(M);
/* Multi Improvement guloso heurístico */
s* ← s;
for k ∈ {1, 2, ..., |M^Sorted|} do
    (m_k, m̂_k) ← M_k^Sorted;
    /* Aplica primeiro movimento disponível */
    s* ← m_k(s*);
    /* Atualiza movimentos independentes */
    for j ∈ {k + 1, ..., |M^Sorted|} do
        (m_j, m̂_j) ← M_j^Sorted;
        /* Elimina movimentos conflitantes */
        if (m_k ∦ m_j) then
            M^Sorted ← M^Sorted \ (m_j, m̂_j);
return s*;
```

Algorithm 3: RDM($f()$, $N()$, $mvCount$, s)

```
/* Iterador de movimentos para vizinhança N */
m^it ← N(s);
/* Aplica movimentos enquanto houver melhoras */
iter ← 0;
s* ← s;
while iter < mvCount do
    /* Gera movimento aleatório na vizinhança */
    m ← random(m^it);
    /* Calcula custo m̂ do movimento m */
    m̂ ← f(m(s)) − f(s);
    /* Somente aceita movimentos de melhora */
    if (m̂ < 0) then
        s* ← m(s*);
        iter ← 0;
    else
        iter ← iter + 1;
return s*;
```

Algorithm 4: Hill Climbing($f()$, $localSearch()$, s)

```
s' ← s;
/* Aplica busca local enquanto houver melhoras */
s* ← localSearch(s');
while f(s*) < f(s') do
    s' ← s*;
    s* ← localSearch(s');
return s*;
```

com grandes dimensões, onde uma exploração exaustiva de busca local não se mostra viável na prática. O Algoritmo 3 descreve o RDM. Por outro lado, o método da descida/subida (do inglês, *Hill Climbing* – HC), é geralmente combinado com a estratégia de melhor vizinho, sendo geralmente nomeada de *Steepest Ascent Hill Climbing*[3]. Ressaltamos que, no escopo do OptFrame, o *Hill Climbing* é visto como uma estratégia genérica de descida/subida, podendo utilizar diversas técnicas de exploração de vizinhança e busca local como forma de operacionalizar a descida/subida (ou seja, é possível combinar HC+BI, ou HC+FI, ou até HC+MIRS). O Algoritmo 4 descreve o *Hill Climbing*.

Uma das meta-heurísticas mais reconhecidas, tanto por sua simplicidade, quanto poder de resolução de problemas desafiadores, é o *Simulated Annealing* (SA) [Kirkpatrick et al. 1983]. A ideia principal é, a partir de uma temperatura corrente, calcular a probabilidade de aceitação de movimentos de melhora ou de piora, de acordo com essa temperatura. À medida em que a temperatura decai, o método se torna um método de busca local, onde somente soluções de melhora são aceitas. Um desafio comum no *Simulated Annealing* é a calibração da temperatura inicial, portanto apresentamos tanto o algoritmo do SA, quando de sua estratégia de calibração (Algoritmos 6 e 5).

[3]https://en.wikipedia.org/wiki/Hill_climbing

3.5 Check Module no OptFrame

A implementação das heurísticas foi marcada por uma grande dificuldade de implementação, característica da implementação de programas complexos em linguagens de baixo nível, dificultada pela natureza complicada do problema em questão. Essa dificuldade de validação de movimentos, acarretou no uso de uma ferramenta muito interessante do OptFrame, chamada CheckModule, que busca resolver exatamente esses problemas. Apontando movimentos e funções com resultados inconsistentes, e gerando um relatório de informações extremamente relevantes para o escopo do Multi Improvement.

Algorithm 5: estimateInitialTemperature($f()$, $N()$, $s_{gen}()$, $\bar{\beta} = 1.1, \gamma = 0.7, SA_{max} = 500, T_0 = 1$)

Result: Temperatura estimada

```
T ← T_0;
s ← s_gen();
Continue ← true;
while (Continue) do
    Accepted = 0;
    for (IterT = 1, 2, ..., SA_max) do
        Generate neighbor s' ∈ N(s);
        Δ ← f(s') − f(s);
        if (Δ < 0) then
            Accepted ← Accepted + 1;
        else
            Let x ∈ [0, 1];
            if (x < e^(−Δ/T)) then
                Accepted ← Accepted + 1;
    if (Accepted ≥ γ · SA_max) then
        Continue ← false;
    else
        T ← β̄ · T;
T_Φ ← T;
return T_Φ;
```

Algorithm 6: BasicSimulatedAnnealing($f()$, $N()$, $s_{gen}()$, $\alpha = 0.98$, $SA_{max} = 500$, $T_0 = T_\Phi$)

```
T ← T_0;
s ← s_gen();
s* ← s;
while T > 0.000001 do
    for (IterT = 1, 2, ..., SA_max) do
        Generate neighbor s' ∈ N(s);
        Δ ← f(s') − f(s);
        if (Δ < 0) then
            s ← s';
            if (f(s) < f(s*)) then
                s* ← s;
        else
            Let x ∈ [0, 1];
            if (x < e^(−Δ/T)) then
                s ← s';
    T ← α · T;
return s*;
```

Os tempos de cada iteração não foram medidos, visto que o objetivo do estudo foi apenas de investigar se a estratégia do Multi Random Selection era válida, e não medir sua performance contra outras abordagens. O ganho real de performance do Multi Improvement é esperado apenas após implementação de programação paralela na fase de busca.

4 RESULTADOS

Como um teste para comparar a funcionalidade do método MIRS com as heurísticas clássicas, nós aplicamos 3 meta-heurísticas diferentes: *Random Descent Method*, *Hill Climbing* com MIRS, e Simulated Annealing. Executamos as 3 heurísticas 10 vezes, sem limite de tempo na instância `rj-0`, e comparamos os resultados obtidos com a solução obtida da geração do valor inicial. O computador utilizado foi um *Intel(R) Core(TM) i7-4820K CPU @ 3.70GHz* com *16 GiB* de memória RAM.

Nessa seção iremos discutir alguns dos resultados mais interessantes obtidos do programa. Na Tabela 1 temos alguns dos resultados das funções de avaliação utilizada, no momento da geração, e no fim da execução de cada heurística.

Método utilizado	Média	Desvio padrão	Coef. de Var.
Solução Inicial	12696449.01	54242.17	0.4%
Simulated Annealing	12683900.00	50754.49	0.3%
Random Descent	12373350.49	58990.80	0.5%
Hill Climbing + MIRS	12340655.15	39691.97	0.4%

Table 1. Resultados do das funções de avaliação após passar por cada um dos processos

Vimos que o método de geração de solução inicial tem os piores resultados, como esperado, seguido de perto pelo Simulated Annealing, observando que o número de iterações foi fortemente restrito no SA para que fosse comparável a métodos simples de busca heurística. As duas melhores soluções foram as encontradas pelo RDM e pelo método de Hill Climbing com a estratégia MIRS, com uma vantagem grande de consistência para a combinação HIll Climbing + MIRS, que apresentou um desvio padrão 20% maior do que qualquer outro método avaliado, mostrando que a a viabilidade de uma heurística com investigação através do MIRS, e a estabilidade que essas abordagens parecem trazer.

Outro resultado interessante do trabalho veio de a ferramenta CheckModule, presente no OptFrame. A ferramente pode ser usada para comparar o tempo de execução de diferentes etapas da execução da meta heurística. Com ela, é possível comparar facilmente as melhores versões de movimentos disponíveis, e fazer as melhores decisões. Com o CheckModule também é possível avaliar a independência de movimentos, oferecendo assim uma grande ajuda na calibração da otimização. A Tabela 2 apresenta um sumário do relatório apresentado pelo Check Module do OptFrame.

5 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho abordou um problema de roteamento de veículos com dados da plataforma loggibud, verificando-se que a grande massa de dados exigia abordagens mais elaboradas do que para a resoluções de problemas de roteamento de veículos na literatura. Apresentamos uma abordagem de clusterização dos dados, para viabilizar o cálculo da matriz de distâncias, bem como apresentamos uma variante do

Solution: Time to clone a solution

#tests	avg(ms)	std(ms)
50	0.0434	0.0066

Constructive: Testing construction of initial solution

#tests	avg(ms)	std(ms)
5	568.4344	9.2632

Evaluators: Testing full evaluate(s) of a solution

#tests	avg(ms)	std(ms)
55	3.5510	0.0120

NS: testing time of cost based on move apply(s) [revCost]

#tests	avg(ms)	std(ms)
25	7.1093	0.0283

NS testing time of cost based on move apply(e, s) [fasterCost]

#tests	avg(ms)	std(ms)
25	7.1096	0.0245

Table 2. Uma parte da saída do CheckModule, dando uma ideia do *custo computacional* de cada etapa do processo de otimização

Multi Improvement, denominada MIRS, que permite exploração aleatória juntamente com a busca por movimentos independentes do Multi Improvement. Também constatamos o benefício do uso do arcabouço de otimização OptFrame, que permite um desempenho compatível com o patamar da linguagem C++, ainda assim permitindo a elaboração de estratégias complexas, mesmo em uma linguagem de baixo nível, com menor esforço e maior segurança do que uma codificação do zero.

Os resultados preliminares aqui presentes apontam para um bom futuro no que tange o Multi Improvement. Resultados mais estáveis e confiáveis, e rápidos, especialmente levando em conta arquiteturas de programação paralela, indicam que o investimento no Multi Improvement tenderá a trazer resultados inovadores. E a implementação do resolvedor em um problema complexo como o proposto pela plataforma loggibud servirá como uma boa prova de conceito na prática.

REFERENCES

David L Applegate, Robert E Bixby, Vašek Chvátal, and William J Cook. 2011. *The traveling salesman problem.* Princeton university press.

Rodolfo Araújo. 2018. Estratégias de exploração de vizinhança com GPU para problemas de otimização. *Programa de Pós-Graduação em Ciências Computacionais - PPG-CComp - UERJ* (2018). https://ccomp.ime.uerj.br/teses-e-dissertacoes/

Geoff Clarke and John W Wright. 1964. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research* 12, 4 (1964), 568–581.

Igor Coelho, Sabir Ribas, Mário Henrique de Paiva Perché, Pablo Munhoz, Marcone Souza, and Luiz Ochi. 2010. OptFrame: a computational framework for combinatorial optimization problems. In *Anais do Simpósio Brasileiro de Pesquisa Operacional.* 1887–1898. http://www.din.uem.br/sbpo/sbpo2010/pdf/72567.pdf

Igor Machado Coelho, Vitor Nazário Coelho, Anderson Zudio, Rodolfo Pereira Araujo, Matheus Nohra Haddad, Pablo Luiz Araújo Munhoz, Breno Maia, Luiz Satoru Ochi, and Marcone Jamilson Freitas Souza. 2020. Microbenchmark Studies in OptFrame: a 10-Year Anniversary. In *Anais do LII Simpósio Brasileiro de Pesquisa Operacional.* https://proceedings.science/sbpo-2020/papers/microbenchmark-studies-in-optframe--a-10-year-anniversary

George B Dantzig and John H Ramser. 1959. The truck dispatching problem. *Management science* 6, 1 (1959), 80–91.

Michael R Garey and David S Johnson. 1979. Computers and intractability. *A Guide to the* (1979).

Fred W Glover and Gary A Kochenberger. 2006. *Handbook of metaheuristics.* Vol. 57. Springer Science & Business Media.

Pierre Hansen and Nenad Mladenović. 2006. First vs. best improvement: An empirical study. *Discrete Applied Mathematics* 154, 5 (2006), 802–817.

Alessandro Souza da Silva Júnior, Igor Machado Coelho, Vitor Nazário Coelho, and Eyder Rios. 2019. Multi Improvement: uma Solução Alternativa para o Problema de Roteamento de Veículos. In *LI SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL, 2019, Limeira. Anais eletrônicos...*, Vol. 2. Galoá. https://proceedings.science/sbpo-2019/papers/multi-improvement--uma-solucao-alternativa-para-o-problema-de-roteamento-de-veiculos

Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. 1983. Optimization by simulated annealing. *science* 220, 4598 (1983), 671–680.

Helena R Lourenço, Olivier C Martin, and Thomas Stützle. 2003. Iterated local search. In *Handbook of metaheuristics.* Springer, 320–353.

Nenad Mladenović and Pierre Hansen. 1997. Variable neighborhood search. *Computers & operations research* 24, 11 (1997), 1097–1100.

Janio Carlos Nascimento Silva, Igor M. Coelho, Ueverton S. Souza, Luiz Satoru Ochi, and Vitor N. Coelho. 2020. Finding the Maximum Multi Improvement on neighborhood exploration. *Optimization Letters* (02 2020). https://doi.org/10.1007/s11590-020-01556-5

Puca Huachi Vaz Penna. 2013. Um Algoritmo Unificado para uma Classe de Problemas de Roteamento de Veículos com Frota Heterogênea. *Programa de Pós-Graduação em Computação - PGC - UFF* (2013). http://www.ic.uff.br/index.php/pt/pos-graduacao/teses-e-dissertacoes

Eyder Rios, Igor M Coelho, Luiz Satoru Ochi, Cristina Boeres, and Ricardo Farias. 2016. A benchmark on multi improvement neighborhood search strategies in cpu/gpu systems. In *2016 International Symposium on Computer Architecture and High Performance Computing Workshops (SBAC-PADW).* IEEE, 49–54.

Eyder Franco Sousa Rios. 2016. Exploração de Estratégias de Busca Local em Ambientes CPU/GPU. *Programa de Pós-Graduação em Computação - PGC - UFF* (2016). http://www.ic.uff.br/index.php/pt/pos-graduacao/teses-e-dissertacoes

Jason Sanders and Edward Kandrot. 2010. *CUDA by example: an introduction to general-purpose GPU programming.* Addison-Wesley Professional.

Marcone Jamilson Freitas Souza. 2008. Inteligência computacional para otimização. *Notas de aula, Departamento de Computação, Universidade Federal de Ouro Preto* (2008). dispon{í}velemhttp://www.decom.ufop.br/prof/marcone/InteligenciaComputacional/InteligenciaComputacional.pdf

Um estudo comparativo de ferramentas de apoio ao teste de mutação para Python

Davi Lobo de Oliveira
Orientador: Auri Marcelo Rizzo Vincenzi
davilobo@estudante.ufscar.br
auri@ufscar.br
Departamento de Computação - Universidade Federal de São Carlos
São Carlos, São Paulo, Brasil

RESUMO

Objetivo: A atividade de teste é uma parte importante no desenvolvimento do software e sua utilização garante maior qualidade por meio da identificação da presença de defeitos que são expostos por essa atividade. Com o crescimento da linguagem Python, o objetivo do estudo é realizar um comparativo entre as ferramentas disponíveis para teste de mutação em Python com intuito de identificar os pontos positivos e negativos de cada ferramenta. **Métodos**: O estudo foi realizado de forma experimental sendo dividido nas seguintes etapas: 1) escolha das ferramentas e do conjunto de programas com testes já existentes, 2) configuração do ambiente para realização dos testes, 3) coleta dos dados obtidos, 4) análise dos dados se baseando em critérios preestabelecidos e por último a conclusão da análise. **Resultados esperados**: A expectativa é que seja possível definir qual ferramenta é melhor para se usar tendo em vista os critérios avaliados. **Conclusões**: A intenção é de se obter informações estatísticas sobre qual seria a ferramenta mais adequada para apoiar o teste de mutação no teste de programas Python.

CCS CONCEPTS

• **Software and its engineering** → **Software notations and tools**; **Software creation and management**; **Software verification and validation**; **Empirical software validation**;

KEYWORDS

teste de software, teste de mutação, linguagem python, automatização de teste de mutação

ACM Reference Format:
Davi Lobo de Oliveira and Orientador: Auri Marcelo Rizzo Vincenzi. 2021. Um estudo comparativo de ferramentas de apoio ao teste de mutação para Python. In *Workshop Loggi'21, 8 de dezembro, 2018, São Paulo, SP*. ACM, New York, NY, USA, 6 pages. https://doi.org/xx.xxxx/xxxxxxx.xxxxxxx

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
WLOGGI'21, 8 de dezembro de 2021,, São Paulo, SP
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/xx.xxxx/xxxxxxx.xxxxxxx

1 INTRODUÇÃO

Devido à crescente utilização de sistemas computacionais em todas as áreas da humanidade, a qualidade é fundamental. Dentre as técnicas de verificação e validação tem-se a atividade de teste que visa a analisar de forma dinâmica o produto, fazendo com que o mesmo falhe, ainda durante o seu desenvolvimento [8].

Dentre os critérios de teste existentes, o teste de mutação é considerado um dos mais eficazes em demonstrar a presença de defeitos, apesar de seu custo elevado. Os principais custos relacionados ao teste de mutação estão relacionados ao grande número de mutantes gerados e que precisam ser executados e posteriormente, analisados quando não for possível distinguir o comportamento entre os mutantes e o programa original. A análise dos mutantes vivos permite melhorar o conjunto de teste ou identificar mutantes semanticamente equivalentes ao programa original, quando não existe caso de teste capaz de diferenciar o seu comportamento [8].

Python é uma das linguagens mais procuradas ultimamente segundo uma pesquisa do Stack Overflow [24] e o índice PYPL [25], entretanto ainda são poucos os estudos comparativos entre as ferramentas de apoio ao teste de mutação para Python.

Este trabalho teve como objetivo conduzir um estudo experimental visando avaliar algumas das ferramentas disponíveis na tentativa de identificar qual(is) seria(m) a(s) melhor(es) opção(ões) dentre as ferramentas estudadas para apoiar o teste de mutação em Python. A avaliação conduzida visa a investigar os aspectos de custo de execução, eficácia e facilidade de uso da ferramenta de forma semelhante ao estudo conduzido por Delahaye e du Bousquet [6].

O restante deste texto está organizado da seguinte forma. Na Seção 2 são descritos a terminologias e os conceitos básicos para a compreensão deste artigo. Na Seção 3 são descritos alguns trabalhos da literatura relacionados com este artigo. Na Seção 4 é apresentada a forma como foi conduzida a avaliação das ferramentas destinadas ao teste de mutação para programas Python, e os resultados obtidos. Finalmente, na Seção 6 são apresentadas as conclusões deste trabalho, bem como as perspectivas de trabalhos futuros.

2 TERMINOLOGIA E CONCEITOS BÁSICOS

2.1 Teste de Software

A atividade de teste está entre as técnicas de verificação e validação e visa a avaliar o produto de software de forma dinâmica expondo suas falhas. Nesse sentido, Myers et al. [23] argumentam que os testes servem apenas para demonstrar a presença de defeitos no software, mas nunca sua ausência. As informações produzidas durante

os testes são de grande importância para as etapas de depuração, manutenção e estimativa de confiabilidade do software.

Visando a reduzir a complexidade dos testes, tal atividade é dividida em fases. Os **testes de unidade** procuram identificar defeitos de lógica e de implementação em cada módulo individual. Os **testes de integração** visam descobrir defeitos entre as interfaces dos módulos, ou seja, na comunicação entre os módulos utilizados para construir a estrutura do programa. Finalmente, os **testes de sistema** busca identificar defeitos de funções e características de desempenho que estejam fora das especificações, além de verificar se todos os componentes do sistema se comportam conforme esperado.

Para avaliar a qualidade da atividade de teste existem as técnicas e critérios de testes. As técnicas de teste são divididas em funcional, estrutural e baseada em defeitos [8]. A **técnica funcional** também conhecida como caixa preta utiliza a especificação do software como base para obter os critérios e requisitos de teste, nessa técnica são verificadas as funções do sistema sem se preocupar com a implementação, os passos principais são a identificação das funções que o software deve realizar e a criação de casos de testes capazes de checar se essas funções estão sendo executadas pelo software. A **técnica estrutural** ou teste caixa branca utiliza o conhecimento da estrutura interna da implementação para a escolha dos casos de teste,ou seja, os critérios e requisitos são obtidos a partir das características de uma implementação específica em teste. A **técnica baseada em defeitos** usa o conhecimento sobre os defeitos típicos cometidos no processo de desenvolvimento do software para estabelecer os critérios e requisitos de testes, a semeadura de defeitos e o teste de mutação são dois dos critérios típicos dessa técnica [8].

2.2 Teste de Mutação

O teste de mutação é considerado um dos critérios mais eficazes em detectar a presença de defeitos, além de ser muito utilizado como modelo de defeitos para avaliar a eficácia de outros critérios de testes [1]. O processo de aplicação do teste de mutação é ilustrado na Figura 1. O objetivo do critério é criar programas com pequenas modificações visando determinar um conjunto de testes capaz revelar as diferenças comportamentais existentes entre o programa original e suas modificações, chamadas de mutantes [11]. Assumindo que cada mutante representa um possível defeito, sendo o comportamento do programa original correto em relação à sua especificação e o comportamento do mutante diferente, conclui-se que o programa original não contém o defeito representado pelo mutante.

Ao executar essas variações de P com o mesmo conjunto T espera-se obter apenas mutantes mortos, ou seja, que foram identificados pelos casos de teste, mas os mutantes podem permanecer vivos quando o conjunto de testes não foi capaz de identificar a mutação e, nesse caso, podem ser adicionados novos casos de teste ao conjunto, ou quando um mutante é equivalente, ou seja, apresenta o mesmo resultado que o programa original para todas as situações e, nesse caso, o mutante pode ser descartado.

É importante ressaltar que determinar se um mutante é equivalente ou não é algo complicado e indecidível do ponto de vista computacional, sendo que, para maioria dos casos é necessário utilizar algumas heurísticas e métodos [20].

Figura 1: Processo de execução dos testes de mutação (extraída da revista Engenharia de Software, Edição 51 Ano 5 – modificada pelo autor)

Uma forma de avaliar a qualidade do teste de mutação é por meio do cômputo do "escore de mutação", conforme Equação 1. O escore de mutação (EM) é dado pela razão do número de mutantes mortos (MM) pelo total de mutantes gerados (TM), excluindo o número de mutantes equivalentes (ME) [10].

$$EM = \frac{MM}{TM - ME} \tag{1}$$

Pode-se observar que o escore de mutação irá variar entre 0 e 1 sendo que quanto mais próximo de 1 mais adequado é o conjunto de casos de teste utilizado em relação ao teste de mutação.

Um dos maiores problemas para uso do teste de mutação é seu alto custo: o número de mutantes gerados mesmo para pequenos programas pode ser muito grande, elevando os tempos de execução e de análise dos mutantes vivos. Pizzoleto et al. [26] conduziram um estudo visando a identificar formas de se minimizar o custo de aplicação do teste de mutação. Espera-se explorar alguns das alternativas no contexto de Python.

2.3 Automatização de Teste de Mutação

Com intuito de garantir maior qualidade e produtividade na atividade de teste é indispensável a utilização de ferramentas automatizadas, essas ferramentas contribuem para a produção de software de melhor qualidade auxiliando pesquisadores e alunos de engenharia de software a compreender conceitos básicos e obter experiência na comparação, seleção e estabelecimento de estratégias de teste [8].

As primeiras ferramentas para teste de mutação surgiram no final da década de 70 e início da década de 80 para a linguagem Fortran e desde então outras ferramentas vêm sendo desenvolvidas para suprir as necessidades dos testadores em diversas linguagens.

Em relação ao teste de mutação, existem ferramentas para diferentes linguagens de programação. Por exemplo, para a linguagem C, existe a ferramenta Proteum/IM [9] com inúmeros recursos para apoio a experimentação e utilização de critérios de mutação alternativos. Para Java, algumas das ferramentas são MuJava [22], Pit [5], Major [18], dentre outras. Para Python também existe um número significativo de ferramentas, algumas delas utilizadas nesse artigo e citadas na Seção 4.3. Nesse sentido, uma questão que se coloca é qual a complementariedade entre as mesmas, sendo esse o objetivo deste estudo considerando as ferramentas destinadas para

Python, que apoiam o teste de mutação em nível de unidade nessa linguagem.

3 TRABALHOS RELACIONADOS

O critério de teste de mutação é algo que vem sendo estudado a algum tempo com várias publicações, entre essas publicações têm-se alguns estudos de análise e comparação das ferramentas disponíveis para teste de mutação, o estudo realizado por Derezińska e Halas [12, 13] em duas partes: a primeira parte em 2014 realizou uma revisão dos operadores que seriam aplicáveis em Python levando em consideração que Python é uma linguagem tipada dinamicamente o que torna a linguagem mais difícil de ser testada se comparado a linguagens fortemente tipadas como Java ou C++, e alguns operadores não fazem sentido nesse tipo de linguagem. A análise desses operadores foi feita usando a ferramenta MutPy [16], e os experimentos mostraram que apenas uma pequena porcentagem dos mutantes gerados eram "incompetentes" e evidenciou o amplo escopo de operadores que o MutPy possui. A segunda parte do estudo em 2015 foram propostas melhorias para o processo de teste de mutação em Python, mostrando os avanços na implementação da ferramenta Mutpy o que reduziu consideravelmente o tempo de execução dos mutantes [12, 13].

Ferreira, Costa e Souza [15] propuseram uma ferramenta para geração de dados de teste de forma automatizada para Python usando o algoritmo Hill Climbing, os testes gerados foram usados em quatro ferramentas: Cosmic-Ray [2], MutMut [17], MutPy [16] e PyMuTester [27], por meio de estudo experimental eles concluíram que a ferramenta foi capaz de matar em média 86% dos mutantes porém ainda é necessário testar a ferramenta com programas mais robustos para fornecer uma ferramenta que possa ser empregada em diferentes cenários [15].

O estudo teve como inspiração o trabalho de Delahaye e Bousquet [7] que realizaram um estudo comparativo entre diferentes ferramentas de apoio ao teste de mutação para Java. Ao todo, oito ferramentas foram comparadas segundo diferentes critérios. Ao final, os autores apresentam o ranking das ferramentas considerando os critérios definidos.

Outro artigo que realizou uma avaliação das técnicas e critérios de teste usando a linguagem Python foi Brito, Andrade e Delamaro [3]. No estudo foi utilizado um conjunto de programas e três ferramentas de teste de mutação, Cosmic-Ray [2], MutMut [17] e MutPy [16]. No estudo, os autores concluíram que um fator primordial para a popularidade e o hábito de utilização dessas ferramentas em cenários de testes reais é o custo computacional e o gerenciamento de recursos para execução das ferramentas.

Com base nas percepções dos resultados dos trabalhos descritos acima, este trabalho apresenta a avaliação do desempenho das ferramentas de mutação MutPy [16], MutMut [17], Mutatest [19] e Cosmic-Ray [2], utilizando um conjunto de programas que implementa algoritmos de estruturas de dados, disponibilizado em repositórios do GitHub [4, 14], e que possuem casos de teste disponíveis. A intenção é avaliar a complementariedade dessas ferramentas, considerando os operadores de mutação para teste de unidade que cada uma implementa, considerando a linguagem Python.

4 AVALIAÇÃO DAS FERRAMENTAS

4.1 Seleção das Ferramentas

Para realizar a seleção das ferramentas foi feito no primeiro momento uma pesquisa de quais ferramentas existiam para a linguagem e onde encontrá-las, após essa etapa foi feito o download e instalação das ferramentas. Inicialmente foram encontradas 8 ferramentas para teste de mutação, uma ferramenta para cobertura de código e um framework de teste. Dentre as ferramentas de mutação foram selecionadas ao final, levando em consideração a documentação disponibilizada, a quantidade de operadores implementados e a facilidade de uso que cada ferramenta possui. A Tabela 1 apresenta a relação das ferramentas de mutação avaliadas e suas respectivas versões (MutPy – MP; MutMut – MM; Mutatest – MT; Cosmic-Ray – CR), além da versão da ferramenta de cobertura e do framework de teste unitário empregado.

Tabela 1: Ferramentas utilizadas

Ferramenta	Versão	Link	Licença	Versão Python
MutPy – MP	0.6.1 - nov/19	https://pypi.org/project/MutPy/	Apache Software License	3.3+
MutMut – MM	2.1.0 - ago/20	https://pypi.org/project/mutmut/	BSD License	3.4+
Mutatest – MT	3.0.2 - mai/20	https://pypi.org/project/mutatest/	MIT License	3.7+
Cosmic-Ray – CR	8.1.0 - mai/21	https://pypi.org/project/cosmic-ray/	MIT License	3.5+
Coverage	6.0.2- out/21	https://pypi.org/project/coverage/	Apache Software License	3.6+
PyTest	6.2.4 - mai/21	https://pypi.org/project/pytest/	MIT License	3.6+

4.2 Seleção dos Programas

A escolha dos programas teve como critério serem implementados em Python e possuírem conjunto de testes disponíveis e que pudessem ser executados pelas ferramentas. Desse modo, o foco do estudo se manteve no desempenho das ferramentas para avaliar os testes disponíveis. Ao todo foram selecionados 40 programas de diferentes tipos como estruturas de dados, algoritmos de ordenação e busca na sua maioria com um conjunto de testes utilizando o framework UnitTest e alguns utilizando o framework PyTest.

Todos os programas foram retirados do GitHub [4, 14].

4.3 Coleta de Dados

O equipamento usado para realizar as atividades foi um notebook com processador i7-7700, 16Gb de memória RAM e o sistema operacional Ubuntu 20.04, para rodar todas as ferramentas foi utilizada a versão 3.8.10 do Python.

A coleta de dados foi realizada da seguinte forma, os conjuntos de testes foram separados em duas pastas, UnitTest e PyTest, essa divisão foi necessária pois os comandos para execução dos testes deve ser parametrizado e buscam pelos testes em locais diferentes. A maioria dos conjuntos de testes foram escritos usando o UnitTest.

Para a análise de cobertura de código foi utilizada a ferramenta Coverage[1] na versão 6.0.2, essa ferramenta monitora quais partes do código foram executadas e quais não foram executadas, sendo possível visualizar a cobertura de comandos e de decisões, a cobertura de código ajuda a descobrir partes do código ainda não executadas pelos testes. A escolha de usar uma ferramenta específica para a

[1] https://pypi.org/project/coverage/

cobertura de código é manter um padrão para o estudo já que nem todas as ferramentas de mutação tem essa funcionalidade.

A Tabela 2 apresenta os dados de tamanho de cada programa utilizado no estudo (coluna sloc). A cobertura de código é usada para avaliar a abrangência do conjunto de teste disponível com os programas utilizados. Como pode ser observado, o conjunto de teste que acompanha cada programa apresenta uma boa qualidade do ponto de vista do teste estrutural, sendo que, em apenas 10 programas dos 40 analisados, a cobertura de comandos e decisões não atinge 100%, e em apenas um a cobertura ficou abaixo de 90%.

Tabela 2: Tamanho e Cobertura de Código dos Programas

Prog.	sloc[2]	statement	miss	exclude	branch	partial	coverage (%)
bfs	57	37	0	0	22	0	100
dfs	34	23	0	0	14	0	100
combinations	36	16	0	0	12	0	100
complement	11	8	0	0	4	0	100
convert-base	15	14	0	0	8	0	100
dijkstras	29	27	0	0	14	0	98
edit-distance	67	44	0	0	26	3	96
factorial	15	5	0	0	4	0	100
fenwick	24	23	0	0	6	0	100
fibonacci	34	11	0	0	6	0	100
find-duplicates	34	29	0	0	16	0	100
fizz-buzz	8	2	0	0	2	0	100
first-mis-pos	37	12	0	0	6	0	100
graph	125	83	0	0	44	1	99
hamming	29	9	0	0	2	0	100
hashmap	102	42	0	0	22	1	98
heap	50	47	0	0	22	0	100
heap-sort	25	24	0	0	18	0	100
linked-list	62	56	0	0	26	2	98
long-comm	27	21	0	0	8	0	100
lru	133	46	0	0	8	0	100
mergesort	24	23	0	0	10	0	100
naive-tree	52	49	0	0	16	0	100
perm-word	13	10	0	0	4	0	100
quicksort	36	17	0	0	6	0	100
rabin-karp	42	17	0	0	12	0	100
radix	21	19	0	0	10	0	100
tree-node	38	36	0	0	22	2	97
trie	32	31	0	0	14	0	100
word-squares	22	19	0	0	8	0	100
binheap	50	38	1	0	18	1	96
bst	206	144	7	0	80	5	93
deque	46	22	0	0	4	0	100
dll	103	82	1	0	36	1	98
graph	57	28	0	0	8	0	100
linked list	73	50	0	0	18	0	100
priorityq	31	17	1	0	2	1	89
queue	30	15	0	0	0	0	100
stack	19	8	0	0	0	0	100
Média	**47,4**	**30,9**	**0,3**	**0,0**	**14,3**	**0,4**	**99,0**
Desvio Padrão	**39,4**	**26,5**	**1,1**	**0,0**	**14,5**	**1,0**	**2,2**

O cálculo da cobertura de comandos é dado pela Equação 2:

$$SCov = \frac{covered\ statements - missing - excluded\ statements}{\#\ of\ statements} \tag{2}$$

A cobertura de comando é calculada subtraindo dos comandos que foram executados as linhas que não foram executadas (*missing*) e os comandos que não serão executados por escolha do programador (*excluded statement*), para identificar um comando *excluded* na ferramenta é necessário usar um comentário `#pragma: nocov`, esse resultado é dividido pelo número total de comandos no programa.

O cálculo da cobertura de decisões é dado pela Equação 3:

$$DCov = \frac{covered\ branches - partial\ branches}{\#\ of\ branches} \tag{3}$$

A cobertura de decisão é calculada por meio do cálculo dos pontos de decisões que foram executadas totalmente (`true` e `false`) menos as decisões executadas parcialmente, ou seja, é feita a diferença entre as decisões que foram testadas todas as possibilidades e as decisões que foram testadas parcialmente. O resultado entre esses valores é dividido pelo número total de decisões.

Para obter o tempo de execução de cada comando foi usado o comando `time` do Linux, esse comando devolve três tempos diferentes: *real* que é o tempo real gasto entre a execução e conclusão do comando que está sendo executado, *user* que é o tempo gasto de processamento e *sys* o tempo usado pelo sistema para executar o comando. Foi usado o tempo real para o cálculo dos programas.

Os comandos usados para a coleta de dados em cada ferramenta estão disponíveis no relatório completo do experimento[3].

Cada programa foi executado pelo terminal conforme comando da ferramenta e gerado um arquivo com o resultado obtido, a Mutpy, Cosmic-Ray geram um relatório em HTML com essas informações, a Mutatest gera um arquivo .RST e a mutmut é capaz de gerar um HTML porém não consta muitas informações no arquivo.

5 RESULTADOS E DISCUSSÃO

Para a análise dos dados são utilizada as informações apresentadas na Tabela 3, geradas pela execução dos programas com as ferramentas de mutação utilizadas e monitoradas pelo comando `time`, conforme mencionado anteriormente. Mais detalhes sobre os dados coletados para os programas podem ser consultados no relatório detalhado do projeto[5].

Tabela 3: Escores e tempo de execução de cada ferramenta

Programa	Escore de Mutação				Tempo de Execução (segundos)			
	MP	MM	MT	CR	MP	MM	MT	CR
bfs	78,1	74,6	81,0	82,8	0,8	9,3	5,4	38,0
binheap	45,0	77,4	93,2	98,2	21,2	45,7	22,7	630,8
bst	77,1	88,0	82,4	85,6	13,0	41,6	8,6	73,5
combinations	66,7	46,0	100,0	100,0	10,4	8,5	2,2	13,7
complement	71,4	80,0	57,1	96,3	5,6	3,0	1,7	70,3
convert-base	93,8	85,3	89,7	91,2	0,5	5,7	3,1	41,1
deque	100,0	100,0	90,0	100,0	0,9	1,0	5,0	3,2
dfs	96,2	79,2	91,4	85,5	0,7	8,0	2,8	37,0
dijkstras	83,3	13,0	0,0	88,6	5,5	9,1	1,8	21,2
dll	82,5	89,0	100,0	83,0	17,5	48,2	12,6	54,0
edit-distance	67,8	80,6	89,7	86,8	72,1	21,5	3,3	1107,6
factorial	57,1	100,0	94,7	97,2	15,3	1,9	3,2	6,9
fenwick-tree	66,7	81,0	94,9	96,5	50,7	10,6	3,6	100,0
fibonacci	100,0	100,0	100,0	100,0	0,5	3,8	2,9	12,0
find-duplicates	78,3	70,0	72,4	71,0	1,8	6,5	3,2	37,1
find-mis-pos	82,4	76,0	85,0	88,1	10,4	6,3	4,6	67,3
fizz-buzz	85,7	100,0	94,3	94,2	5,4	3,8	2,9	31,2
graph	66,7	100,0	83,3	100,0	1,1	2,0	3,6	4,5
graph-cycles	61,5	57,4	60,0	62,2	16,4	8,8	1,9	53,6
hamming	80,0	72,7	83,3	96,6	5,3	0,5	1,2	68,6
hashmap	100,0	97,6	81,3	93,6	0,6	7,1	3,7	143,0
heap	59,4	84,1	82,1	86,6	101,7	14,4	4,1	445,0
heapsort	66,7	86,1	73,7	85,7	0,6	6,2	3,1	119,0
linked-list	77,8	61,5	72,2	93,2	20,6	19,7	3,4	47,4
linked-list-pytest	84,0	93,5	100,0	95,8	4,8	23,6	13,4	35,3
longest-commom	90,9	100,0	87,0	86,7	0,4	2,9	1,9	11,5
lru	83,3	76,5	66,7	73,1	0,5	5,9	2,0	7,5
mergesort	81,3	93,8	96,7	96,0	10,6	2,9	2,3	12,2
permutation-word	57,1	100,0	95,7	100,0	15,5	1,8	2,4	26,2
priorityq	25,0	77,8	75,0	84,6	1,1	5,9	5,8	8,0
queue	50,0	100,0	75,0	100,0	1,3	2,7	2,6	3,4
quicksort	60,0	88,2	85,7	88,3	0,5	3,8	2,7	16,0
rabin-karp	94,7	87,2	85,2	86,1	0,7	6,9	4,4	66,0
radixsort	55,6	76,5	78,6	85,7	10,4	3,8	3,3	43,7
stack	0,0	100,0	50,0	0,0	0,0	1,2	1,8	1,9
topological-sort	28,2	24,1	26,9	15,3	1,3	10,0	2,4	24,0
tree	92,0	6,7	9,1	100,0	10,7	2,9	1,9	9,2
trie	0,0	22,7	0,0	10,3	0,4	4,0	1,7	9,0
weighted-graph	25,6	27,8	51,7	28,6	11,3	8,9	2,5	25,3
word-squares	50,0	100,0	88,9	87,5	10,3	2,7	1,5	8,5
Média	68,0	76,8	75,6	82,5	11,5	9,6	4,1	88,3
Desvio Padrão	25,2	25,8	26,1	25,0	20,2	11,6	4,0	202,6

[3]https://docs.google.com/document/d/17Hx58wMXmgBY_KpTE_TAZQydmND51OvYWzExQv-MMcc/edit

5.1 Cosmic-Ray – CR

Como mostrado na Tabela 3 a maior média de escore de mutação obtida foi da Cosmic-Ray com 78,36%, porém analisando o tempo na mesma tabela percebe-se que a ferramenta teve o maior tempo de execução e o que contribuiu para isso foi a quantidade de mutantes gerados em comparação as demais ferramentas, essa diferença pode ser vista na Figura 3. Algumas desvantagens da ferramenta quando comparada às outras é a dificuldade da criação do arquivo de configuração .TOML e o consumo de recursos do sistema, o que em alguns casos chegou a causar o travamento do sistema, sendo assim, se por um lado ela conseguiu detectar um número maior de mutantes, por outro apresentou desvantagens que são fatores importantes quando queremos escolher uma ferramenta de teste.

De toda forma, do ponto de vista do escore de mutação, conforme pode ser observado no gráfico da Figura 2, ela foi a ferramenta mais consistente.

5.2 MutMut – MM

A MutMut foi desenvolvida com foco na facilidade de uso e ela cumpre esse objetivo, para o conjunto de programas utilizado ela gerou 1371 mutantes desses 990 foram mortos, o tempo de execução foi de aproximadamente 6 minutos e 38 segundos ficando atrás apenas da Mutatest, a desvantagem da ferramenta é não informar qual o caso de teste que conseguiu detectar o mutante, no relatório temos apenas os mutantes que sobreviveram.

Outro ponto importante é em relação a como os testes são feitos, as mutações ocorrem direto no código original e após a execução dos casos de teste é possível salvar o código fonte com a modificação, por padrão elas são retiradas voltando o código para o original, o problema em realizar os testes dessa maneira é se ocorrer algum erro na execução e o código fonte permanecer com a modificação realizada causando um defeito no programa.

5.3 Mutatest – MT

Essa foi a ferramenta mais rápida para executar os testes como podemos notar ao analisar a Figura 4, a possibilidade de escolher qual o modo de execução para o conjunto de teste é o diferencial quando comparado às outras ferramentas, sendo possível rodar todo o conjunto de teste em todas as mutações ou passar para a próxima mutação quando um caso de teste detectou ou não o mutante, porém assim como a MutMut temos um conjunto de operadores de mutação limitado.

5.4 MutPy – MP

A MutPy possui o maior conjunto de operadores de mutação das ferramentas apresentadas são 27 operadores sendo 7 operadores experimentais, o relatório produzido pela ferramenta é o mais organizado e de fácil entendimento entre as quatro participantes do estudo, essas características facilitam o seu uso, se compararmos o tempo de execução das ferramentas a MutPy fica atrás apenas da Cosmic-Ray, com relação ao número de mutantes gerados e mortos foi a de pior desempenho mas o que chama a atenção para essa ferramenta é que os operadores de mutação utilizados buscam gerar mutantes que testam características específicas de Python, ao invés de mutações mais genéricas como os gerados pelas outras ferramentas.

Algo importante a se destacar é que todas as ferramentas apresentaram um desempenho ruim nos programas que envolviam grafos.

Figura 2: Média de escores de mutação por ferramenta

Figura 3: Média de mutantes gerados por ferramenta

Figura 4: Média dos tempos de execução das ferramentas

6 CONCLUSÃO

Este trabalho teve como objetivo realizar a avaliação de ferramentas de teste de mutação para python destacando seus pontos positivos e negativos,para isso foram avaliadas quatro ferramentas utilizando um conjunto composto por 40 programas de tipos diferentes.

Os resultados demonstraram que é difícil afirmar a existência de uma ferramenta que é a melhor em todos os aspectos, existe uma boa documentação para cada ferramenta e cada uma possui pontos positivos e negativos, a MutPy apesar de não ter obtido os melhores resultados é uma ferramenta que possibilita analisar quais foram os mutantes que sobreviveram, quais casos de teste foram usados, fornece cobertura de código e não possui um alto custo de execução. A Mutatest apesar de ser uma ferramenta rápida e que possibilita critérios diferentes para executar os testes têm como desvantagem operadores mais genéricos, assim como a MutMut que apesar de ser fácil de utilizar não fornece muitas informações no seu relatório gerado, já a Cosmic-Ray tem um custo de execução elevado mas das ferramentas utilizadas é a que tem recebido mais atualizações e correções, além de ser extensível por meio de plug-ins.

Como trabalhos futuros está prevista uma comparação entre as ferramentas utilizando os conjuntos de teste adequados a cada uma de forma individual e, cruzando a cobertura e o escore de mutação desse conjunto com as demais ferramentas. O problema que se tem para realizar essa atividade é identificar quais casos de teste matam quais mutantes. Essa informação não está disponível em todas as ferramentas o que dificulta a análise. No caso da MutPy esse relatório está disponível e já se iniciou a coleta de dados comparando qual a capacidade dos casos de teste que matam mutantes da MutPy em matar os mutantes das demais ferramentas. O intuito dessa comparação é observar se existe alguma diferença entre os conjuntos e otimizar o número de casos de teste, além de identificar o aspecto complementar entre as ferramentas.

Outra possibilidade é avaliar os testes gerados de forma automatizada por uma ferramenta chamada Pynguin [21] que está em desenvolvimento e utiliza o framework PyTest para gerar os testes de unidade.

Considerando os programas utilizados, pretende-se gerar um conjunto de teste 100% adequado ao teste de mutação para cada uma das ferramentas estudadas, determinando os mutantes equivalentes de forma manual e, desse modo, construir uma base de programas Python que possa ser utilizada em experimentação.

Finalmente, pretende-se utilizar a ferramenta mais adequada no programa Loggi-Bud[4], avaliando a qualidade do conjunto de teste disponível.

ACKNOWLEDGMENTS

Loggi Tecnologia pelo apoio financeiro.

REFERÊNCIAS

[1] J. H. Andrews, L. C. Briand, and Y. Labiche. 2005. Is Mutation an Appropriate Tool for Testing Experiments?. In *Proceedings of the 27th International Conference on Software Engineering* (St. Louis, MO, USA) *(ICSE '05)*. Association for Computing Machinery, New York, NY, USA, 402–411. https://doi.org/10.1145/1062455.1062530

[2] Austin Bingham. 2021. Cosmic Ray: mutation testing for Python. (Nov. 2021). https://github.com/sixty-north/cosmic-ray

[3] Renata O. Brito, Stevão A. Andrade, and Marcio E. Delamaro. 2020. Uma avaliação de técnicas e critérios de teste de software para a linguagem de programação Python. In *XXVII Simpósio Internacional de Iniciação Científica e Tecnológica da USP – SIICUSP'2020*. Edusp, São Carlos, SP.

[4] Clair3st. [n.d.]. *Data Structures*. Retrieved 8 de Novembro, 2021 from https://github.com/clair3st/Data-Structures

[5] Henry Coles. 2015. PITest: real world mutation testing. Disponível em: http://pitest.org/. Acesso em: 04/07/2016. bibtex*[howpublished=Página Web].

[6] M. Delahaye and L. du Bousquet. 2013. A Comparison of Mutation Analysis Tools for Java. In *Quality Software (QSIC), 2013 13th International Conference on*. 187–195. https://doi.org/10.1109/QSIC.2013.47

[7] Mickaël Delahaye and Lydie du Bousquet. 2013. A Comparison of Mutation Analysis Tools for Java. In *2013 13th International Conference on Quality Software*. 187–195. https://doi.org/10.1109/QSIC.2013.47

[8] M. E. Delamaro, J. C. Maldonado, and M. Jino. 2016. *Introdução ao Teste de Software* (2 ed.). Elsevier, Rio de Janeiro, RJ.

[9] M. E. Delamaro, J. C. Maldonado, and A. M. R. Vincenzi. 2000. Proteum/IM 2.0: An Integrated Mutation Testing Environment. In *Mutation 2000 Symposium*. Kluwer Academic Publishers, 91–101. event-place: San Jose, CA.

[10] R.A. DeMillo. 1980. Mutation analysis as a tool for software quality assurance. Technical report. *Computer* (1980).

[11] R.A. DeMillo, R.J. Lipton, and F.G. Sayward. 1978. Hints on Test Data Selection: Help for the Practicing Programmer. *Computer* 11, 4 (1978), 34–41. https://doi.org/10.1109/C-M.1978.218136

[12] Anna Derezinska and Konrad Hałas. 2014. Analysis of Mutation Operators for the Python Language. *Advances in Intelligent Systems and Computing* 286 (01 2014), 155–164. https://doi.org/10.1007/978-3-319-07013-1_15

[13] Anna Derezinska and Konrad Hałas. 2015. Improving Mutation Testing Process of Python Programs. *Advances in Intelligent Systems and Computing* 349 (01 2015), 233–242. https://doi.org/10.1007/978-3-319-18473-9_23

[14] exterkamp. [n.d.]. *Python Data Structures*. Retrieved 8 de Novembro, 2021 from https://github.com/exterkamp/Python-Data-Structures

[15] Matheus Ferreira, Lincoln Costa, and Francisco Carlos Souza. 2020. Search-based Test Data Generation for Mutation Testing: a tool for Python programs. In *Anais da IV Escola Regional de Engenharia de Software* (Evento Online). SBC, Porto Alegre, RS, Brasil, 116–125. https://doi.org/10.5753/eres.2020.13722

[16] Philipp Hossner, Konrad Hałas, Steven Myint, and Andreas Mueller. 2021. MutPy: a mutation testing tool for Python 3.x source code. (Nov. 2021). https://github.com/mutpy/mutpy

[17] Anders Hovmöller. 2021. Mutmut: a Python mutation testing system. https://hackernoon.com/mutmut-a-python-mutation-testing-system-9b9639356c78

[18] René Just, Franz Schweiggert, and Gregory M. Kapfhammer. 2011. MAJOR: An efficient and extensible tool for mutation analysis in a Java compiler. In *Proceedings of the International Conference on Automated Software Engineering (ASE)*. 612–615.

[19] Evan Kepner. 2011. Mutatest: Python mutation testing. (Nov. 2011). https://github.com/EvanKepner/mutatest

[20] Marinos Kintis, Mike Papadakis, Andreas Papadopoulos, Evangelos Valvis, and Nicos Malevris. 2016. Analysing and Comparing the Effectiveness of Mutation Testing Tools: A Manual Study. In *2016 IEEE 16th International Working Conference on Source Code Analysis and Manipulation (SCAM)*. 147–156. https://doi.org/10.1109/SCAM.2016.28

[21] Stephan Lukasczyk, Florian Kroiß, and Gordon Fraser. 2020. Automated Unit Test Generation for Python. In *Proceedings of the 12th Symposium on Search-based Software Engineering (SSBSE 2020, Bari, Italy, October 7–8) (Lecture Notes in Computer Science, Vol. 12420)*. Springer, 9–24. https://doi.org/10.1007/978-3-030-59762-7_2

[22] Y.-S. Ma, J. Offutt, and Y. R. Kwon. 2005. MuJava: an automated class mutation system: Research Articles. *STVR – Software Testing, Verification and Reliability* 15, 2 (2005), 97–133. https://doi.org/10.1002/stvr.v15:2 bibtex*[location=Chichester, UK, UK;publisher=John Wiley and Sons Ltd.].

[23] Glenford J. Myers, Corey Sandler, and Tom Badgett. 2011. *The Art of Software Testing* (3rd ed.). Wiley Publishing.

[24] Stack Overflow. [n.d.]. *2021 Developer Survey*. Retrieved 8 de Novembro, 2021 from https://insights.stackoverflow.com/survey/2021#most-loved-dreaded-and-wanted-database-want

[25] Stack Overflow. [n.d.]. *PYPL*. Retrieved 8 de Novembro, 2021 from https://pypl.github.io/PYPL.html

[26] Alessandro Pizzoleto, Fabiano Ferrari, Jeff Offutt, Leo Fernandes, and Márcio Ribeiro. 2019. A Systematic Literature Review of Techniques and Metrics to Reduce the Cost of Mutation Testing. *Journal of Systems and Software* 157 (07 2019). https://doi.org/10.1016/j.jss.2019.07.100

[27] Michael Teo. 2021. PyMuTester: Python Mutant Testing. (Nov. 2021). https://miketeo.net/blog/projects/python-mutant-testing-pymutester

[4] https://github.com/loggi/loggibud

Especificação e Verificação Formal em Aplicações de Logística

Victor Ribeiro, Marcel Oliveira
Universidade Federal do Rio Grande do Norte
Natal, Brasil
cstavictor@gmail.com,marcel@dimap.ufrn.br

1 INTRODUÇÃO

O planejamento, implementação, coleta, armazenamento e distribuição de encomendas exige das empresas de logística uma otimização constante de todas essas tarefas a fim de garantir eficácia, eficiência e segurança da operação. Isto porque uma logística bem planejada representa entregas mais rápidas, reduzindo estoques e custos operacionais, e aumentando produtividade e giro de mercadorias. Desta forma, [6] acredita que a logística passou a ser sinônimo de redução de custos e melhoria de serviços, observando a relação entre clientes e fornecedores, fornecendo um loop mais confiável entre todos, sendo portanto, um sistema de coordenação sistemática com objetivos comuns.

Para atingir este objetivo, faz-se necessário a utilização de um serviço de roteirização que consiste em um problema de otimização combinatória para a definição de rotas para o transporte de encomendas, tendo como resultado a melhor solução para o transportador. Porém, a definição das melhores rotas vai além da tarefa de encontrar a rota mais rápida, uma vez que informações relevantes devem ser também consideradas, tais como ambiente do tráfego, o congestionamento do tráfego e a segurança do tráfego [8], de forma a garantir a eficácia, eficiência e segurança da operação de transporte. Embora as várias combinações de rotas possam ser testadas manualmente, ou até mesmo através da utilização de ferramentas, testes exaustivos podem não ser viáveis dada a imensa quantidade de possibilidades de rotas resultantes de um processo de roteirização. Além disso, as próprias ferramentas não podem oferecer total garantia de que a rota "vencedora" seja, de fato, a melhor rota a ser executada. Neste contexto, surgem os Métodos Formais, uma área da engenharia de software que visa, utilizando-se de métodos fortemente embasados em formalismos matemáticos, visam garantir o correto funcionamento de sistemas complexos e críticos como sistemas da área aeronáutica, área médica, área metro-ferroviária, financeira e logística. Ao construir um modelo matematicamente rigoroso de um sistema complexo, é possível verificar as propriedades do sistema de maneira mais completa do que os testes empíricos. A sua importância pode ser exemplificada pelas regulamentações de certificação DO-178C e DO-278A que fazem parte do corpo de regulamentações da área aeronáutica e exigem a utilização destes métodos no desenvolvimento de sistemas aeronáuticos. Outro exemplo da importância estratégica da utilização deste método pode também ser encontrada no padrão IEC 61508 (Segurança Funcional de Sistemas Relacionados à Segurança Elétrica / Eletrônica / Eletrônica Programável), um padrão internacional publicado pela International Electrotechnical Commission que consiste em métodos sobre como aplicar, projetar, implantar e manter sistemas de proteção automáticos chamados de sistemas críticos, cujos nível mais alto de certificação, SIL4, recomenda fortemente a utilização de métodos formais. Este projeto tem como objetivo investigar o potencial de utilização de técnicas de Métodos Formais para verificar formalmente as soluções propostas por um processo de roteirização, fortalecendo assim, os resultados obtidos.

2 TRABALHOS RELACIONADOS

Ratke et al [5] apresentaram uma estratégia para planejar uma rota com o menor custo utilizando-se de Constraint Satisfaction Problem, uma técnica também chamada de programação por restrições, que tem a capacidade de tornar o desenvolvimento de programação mais natural para alguns tipos de problemas devido a fatores como o uso de ferramentas que tornam as soluções mais automatizadas. Seguindo outra abordagem, Saddiq et al [7], apresenta um modelo formal em VDM de um serviço de logística inteligente capaz de identificar demandas de clientes, produtos e gestão de estoque para desenvolver um sistema de logística inteligente. Este modelo é então verificado utilizando as ferramentas do método em questão. No entanto, em seu trabalho, eles apresentam apenas a verificação de consistência da especificação. Uma das técnicas que utilizam álgebras de processo foi apresentada em [3], onde a verificação de modelos, uma técnica que permite a verificação de sistemas concorrentes, foi utilizada para verificar a segurança de sistemas logísticos. No entanto, este trabalho não apresenta os detalhes da aplicação desta técnica ao contexto de logística e apenas apresenta superficialmente a técnica de verificação de modelos. Neste projeto, pretendemos aprofundar estas análises, possivelmente apresentando resultados práticos iniciais da aplicação de várias técnicas de métodos formais. Nosso foco, no entanto, serão as técnicas baseadas em modelo, como em [7], técnicas baseadas em álgebras de processo, como em [3], e a aplicação de cálculo de refinamentos.

3 DESIGN BY CONTRACT

Design by contract (DBC) [4] é um método de desenvolvimento de software que se baseia no conceito de "contratos"estabelecidos entre as classes e seus clientes. Por um lado, o cliente deve garantir certas condições antes de chamar um método definido pela classe (pré-condição). Em troca, a classe garante que certas propriedades serão atendidas no retorno da chamada (pós-condição). Esta noção de pré- e pós-condições não é uma novidade lançada pelo DBC, uma vez que foi inicialmente publicada por Hoare em seu artigo de 1969 sobre verificação formal [1]. A novidade no DBC é que ele torna esses contratos executáveis, uma vez que eles são definidos no próprio código fonte do programa e são tratados pelo compilador.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
Workshop Virtual Loggi, 8 de dezembro de 2021, On-line
© 2021 Copyright held by the owner/author(s).

Desta forma, qualquer violação do contrato que ocorra enquanto o programa está em execução pode ser detectado imediatamente.

O uso de DBC fornece boa documentação para módulos dos sistemas tais como classes e interfaces Java. Para cada método de uma classe ou interface, o contrato define a pré-condição que o cliente deve atender e a pós-condição que a classe ou interface deve oferecer ao cliente. Desta forma, DBC não se trata apenas de código, mas também oferece uma melhor documentação, a qual define mais precisamente as características dos módulos do que apenas comentários de programa ou javadoc. Por um lado, uma especificação de contrato DBC é mais abstrata do que o código pois não define um algoritmo em detalhes, mas pode informar precisamente o que é presumido e o que deve ser alcançado. Esta definição do contrato, diferente do que oferece um simples comentário java ou javadoc, pode ser verificada mecanicamente.

4 JML

JML (Java Modeling Language) [2] é uma linguagem de especificação de contratos formais para Java. Desta forma, ela permite a especificação tanto das interfaces sintáticas dos métodos quanto seus comportamentos. Desta forma, JML é bem adequada para documentar decisões de design detalhadas sobre código Java.

JML combina a praticidade da abordagem DBC com a expressividade e formalidade de linguagens de especificação orientadas a modelos, usando a sintaxe de expressões de Java para escrever os predicados usados em asserções, como pré-condições, pós-condições e invariantes. Esta abordagem trás consigo a vantagem de facilitar o seu uso pelos próprios programadores. No entanto, JML extende as expressões Java com várias outras estruturas necessárias para aumentar a expressividade das expressões como, por exemplo, quantificadores lógicos.

JML é o único que foi projetado para ser usado com uma ampla gama de ferramentas. Estas ferramentas suportam DBC, verificação de asserção em tempo de execução, descoberta de invariantes, verificação estática, navegação de especificações e até verificação formal usando provadores de teoremas. Apesar de toda a sua capacidade, neste projeto, pretendemos utilizar um pequeno subconjunto que pode ser usado como uma linguagem DBC para Java.

5 CONSIDERAÇÕES FINAIS

No artigo conjunto "Uma Abordagem Multidimensional para Roteirização no Território Inteligente"apresentamos uma abordagem multimensional para roteirização que considera informações providas pelo território inteligente. Tal abordagem correlaciona os dados de rotas com dados de outros domínios da cidade que podem ter impacto na escolha da rota. Naquele projeto, ainda pretende-se aprimorar a integração da abordagem multidimensional proposta para avaliação de rotas com a plataforma SGeoL,a qual consistirá no desenvolvimento de uma aplicação sobre a SGeoL que possibilite aos usuários não apenas visualizarem as rotas geradas e avaliadas, como também possam configurar as ponderações dadas às dimensões consideradas de acordo com seus interesses. Outra linha de trabalho futuro está relacionada à incorporação de outras dimensões do território inteligente para avaliação das rotas.

No contexto de "Especificação e Verificação Formal em Aplicações de Logística", utilizaremos JML para aplicar a metodologia DBC na aplicação construída como forma de aumentar a confiabilidade dos resultados.

REFERÊNCIAS

[1] C. A. R. Hoare. 1969. An Axiomatic Basis for Computer Programming. *Commun. ACM* 12, 10 (oct 1969), 576–580. https://doi.org/10.1145/363235.363259

[2] Gary Leavens and Yoonsik Cheon. 2004. Design by contract with JML. (07 2004).

[3] Robert Meolic, Tatjana Kapus, and Zmago Brezočnik. 2000. Model Checking: A Formal Method for Safety Assurance of Logistic Systems.

[4] Bertrand Meyer. 1992. Applying "Design by Contract". *Computer* 25, 10 (1992), 40–51. https://doi.org/10.1109/2.161279

[5] Cláudio Ratke, Hélcio Hoffmann, and Hannelore Nehring. 2015. Routing of Vehicles Using CSP: Case Sudy. In *2015 Ninth International Conference on Complex, Intelligent, and Software Intensive Systems*. 250–253. https://doi.org/10.1109/CISIS.2015.34

[6] Paulo Roberto Ambrosio Rodrigues. 2007. *Introdução aos sistemas de transporte no Brasil e à logística internacional*. Aduaneiras, São Paulo.

[7] Saima Saddiq, Nazir Ahmad Zafar, and Farhan Ullah. 2017. Formal modeling of smart logistics monitoring. In *2017 1st International Conference on Electronics, Materials Engineering and Nano-Technology (IEMENTech)*. 1–6. https://doi.org/10.1109/IEMENTECH.2017.8076997

[8] Eiichi Taniguchi, Russell G. Thompson, Tadashi Yamada, and Ron van Duin. 2001. *City logistics: Network modeling and intelligent transport systems*. Emerald Publishing, United Kingdom.

Middleware Específico de Domínio para Aplicações e Serviços de Entregas Urbanas

Marcos Vinícius S. Oliveira
Instituto de Informática
Universidade Federal de Goiás
Goiânia, Goiás, Brasil
marcosoliver@discente.ufg.br

Fábio M. Costa
Instituto de Informática
Universidade Federal de Goiás
Goiânia, Goiás, Brasil
fmc@inf.ufg.br

RESUMO

Nos últimos anos tornou-se evidente o aumento da complexidade no domínio de logística de entregas urbanas devido ao crescimento do e-commerce. A literatura aborda o problema da logística de entregas urbanas dando maior ênfase à atividade de entrega em si. Porém um outro importante gargalo observado durante a revisão da literatura e análise do domínio é a atividade de coleta dos pacotes até o centro de distribuição. Este trabalho propõe a otimização da atividade de coleta de pacotes através da associação dos pontos de coleta às rotas dos entregadores. Diante da complexidade do cenário faz-se necessária a utilização de uma solução customizável por um especialista de domínio. Por isso, propomos a utilização de middleware específico de domínio orientado a modelos, na qual o especialista especifica um modelo de coleta com base no qual o Middleware associa a coleta à rota mais viável determinada por uma função de utilidade.

Palavras-Chave

logística urbana, delivery, última milha, otimização

1 Introdução

O domínio de entregas urbanas é caracterizado por sua alta complexidade, que está cada vez mais acentuada devido ao e-commerce, como pode ser observado no crescimento registrado em setembro de 2021, com um aumento de 24,21% em relação ao mesmo mês do ano anterior[1]. Algoritmos são utilizados para determinar as rotas mais eficientes na realização das entregas; pacotes possuem tamanhos, prioridades e fragilidades diferentes; do envio até o recebimento, os pacotes passam por diversos ambientes, dentre eles centros de distribuição, rodovias, centros urbanos e domicílios [4].

Os pacotes estarem armazenados no centro de distribuição é requisito fundamental para que todo o processo de entrega de pacotes seja realizado. Assim, é evidente que a atividade de coleta de pacotes e sua entrega no centro de distribuição pode se tornar um gargalo e comprometer a eficiência do processo caso não seja devidamente estudada, planejada e amparada pela tecnologia.

Diante da alta complexidade do domínio, este trabalho propõe uma solução baseada em middleware específico de domínio [1] para associar as coletas de pacotes às rotas de entrega em execução. Como as coletas de pacotes podem ser realizadas de diversas maneiras diferentes, propomos uma linguagem de modelagem para que um especialista de domínio possa especificar um modelo de coleta de pacotes capaz de atender às necessidades do negócio.

Considerando que a maior parte do problema se passa no ambiente urbano, a implantação de um middleware para este domínio deve considerar o uso de dispositivos heterogêneos como fontes de dados, sendo as cidades inteligentes um paradigma importante para lidar com essa heterogeneidade e permitir o fornecimento de serviços inteligentes. O conceito de Cidades Inteligentes utiliza diversos recursos heterogêneos do cenário urbano como fontes de dados que, após processados, são utilizados para otimizar e automatizar atividades como controle de tráfego, iluminação pública e gestão de energia. Diante da heterogeneidade dos recursos, a utilização de Middleware Específico de Domínios Orientados a Modelos facilita e uniformiza o desenvolvimento de aplicações para Cidades Inteligentes, onde modelos específicos de domínio são interpretados por um Middleware para guiar o acesso e

[1] De acordo com o índice MCC-ENET, desenvolvido pela Neotrust em parceria com o Comitê de Métricas da Câmara Brasileira da Economia Digital.

manipulação dos recursos urbanos que serão utilizados nas aplicações [3].

Neste contexto, propomos o uso de middleware específico de domínio para resolver de forma flexível o problema de associar as coletas de pacote às rotas de entrega. A proposta incluiu a implementação de uma linguagem de modelagem para definir modelos de coleta, juntamente com a definição do conhecimento específico de domínio necessário ao middleware para interpretar os modelos de coletas e por meio do uso de uma função de utilidade, associar cada coleta à rota de entrega mais eficiente. O trabalho encontra-se em andamento e seu estado atual é descrito neste artigo.

O artigo está estruturado como se segue. A Seção 2 discorre sobre a revisão da literatura que forneceu o embasamento teórico para o desenvolvimento do trabalho. A Seção 3 apresenta a visão geral da proposta e as partes que atuam em conjunto para o seu funcionamento. A Seção 4 apresenta o protótipo implementado, que posteriormente será utilizado na implementação e avaliação da Linguagem de Modelagem e na definição do conhecimento específico de domínio. Por fim, a Seção 5 apresenta a conclusão e os trabalhos futuros.

2 Revisão da literatura

Apesar de usufruirmos da logística de entregas diariamente, muitas vezes não percebemos a complexidade e os mecanismos envolvidos no processo de transporte de um pacote de sua origem até o seu destino. O trabalho de Cardenas (2018) discorre sobre como as cadeias de suprimentos, logistica e frete têm enfrentado desafios cada vez mais complexos apresentados por transições nas estruturas econômicas, urbanização, design da cidade e sistemas de transporte, bem como por as externalidades associadas às atividades logísticas em áreas urbanas [4].

No domínio de logística de entregas, o conceito de última milha se refere à etapa final do processo, mais especificamente a saída do pacote do centro de distribuição até o local de entrega. Esta última etapa ocorre majoritariamente nos centros urbanos e pode fazer uso da tecnologia da informação para acessar os recursos da cidade de forma mais inteligente [3], conceito este presente no paradigma de cidades inteligentes no domínio da mobilidade inteligente.

Para dar suporte a aplicações que consomem recursos heterogêneos e ao mesmo tempo permitir que especialistas de domínio definam aplicações, middleware específico de domínio orientados a modelos são capazes de abordar estes dois problemas de forma eficiente como proposto por (COSTA et al., 2017).

A orientação à modelos dos middlewares específicos de domínio é possível através da definição de uma linguagem de modelagem capaz de definir os modelos que serão interpretados para a geração de scripts de controle. Em seu trabalho é definida uma linguagem de modelagem específica de domínio para especificação de modelos para aplicações de gerenciamento de energia em Smart Grids [2].

3 Visão geral da proposta

A Figura 1 apresenta uma visão geral da proposta. Tendo o middleware como componente central. Como entrada, o Middleware recebe uma lista de rotas de entrega e uma lista de pontos de coleta. A linguagem de modelagem é usada para definir modelos de coleta, que são passados para o Middleware, o qual também possui uma especificação do Conhecimento de Domínio necessário para a interpretação dos modelos.

Todas essas peças são utilizadas pelo Middleware para, com o uso de uma função de utilidade, associar os pontos de coleta às rotas de entrega.

Figura 1. Visão Geral da Proposta

3.1 Linguagem de Modelagem

A flexibilidade na construção de aplicações pelo especialista de domínio é possível pela definição de uma linguagem de modelagem por meio da qual pode-se definir modelos para execução das coletas de pacotes. A linguagem permite definir um conjunto de restrições que determinam a forma como as coletas devem ser realizadas.

As restrições são compostas por: um nome descritivo, como, Distância, Tempo, Desvio; um peso, que determina o quão importante a restrição é em relação ao cenário geral;

e um valor máximo, que funciona como um teto para a restrição. Por exemplo, se na restrição Desvio for definido um peso 10 e o valor máximo de 5km, qualquer rota que possua um desvio maior que o teto automaticamente será desconsiderada como uma rota viável para realizar a coleta, e os valores que não ultrapassam o teto serão multiplicados pelo peso definido.

3.2 Middleware Específico de Domínio Orientado a Modelos

No contexto de cidades inteligentes, middleware específico de domínio é utilizado para preencher a lacuna entre os recursos heterogêneos disponíveis no centro urbano e as aplicações, uniformizando a comunicação.

A orientação a modelos abstrai a complexidade no desenvolvimento das aplicações. Assim, o próprio especialista do domínio é capaz de definir suas aplicações através de blocos básicos de construção.

O middleware específico de domínio interpreta modelos em tempo de execução e gera scripts de controle, coordena a entrega de serviços através de uma API independente, e comunica com as estruturas e controladores subjacentes para entregar os serviços no domínio.

Neste trabalho, os scripts de controle gerados pelo Middleware Específico de Domínio associam pontos de coleta a rotas de entrega utilizando uma função de utilidade que utiliza as restrições definidas no modelo pelo especialista de domínio.

Uma função de utilidade neste contexto, é definida da seguinte forma:

$$utilidade = \sum_{i=1}^{n} X_i * pX_i$$

Onde X_i é o valor de uma propriedade da rota, como, a distância, o tempo, ou desvio; pX_i é o peso da propriedade definido na restrição; e n é a quantidade de restrições definidas no modelo.

Assim, utilidade de uma rota em relação a um ponto de coleta é dada pelo somatório do valor de cada propriedade da rota multiplicado pelo seu respectivo peso.

4 Protótipo e Cenário de Testes

Esta seção apresenta o gerador de rotas que será utilizado na avaliação da linguagem de modelagem e na definição do conhecimento específico de domínio. A seção apresenta também um cenário de teste que contém uma primeira implementação dos algoritmos que serão implementados posteriormente no middleware.

4.1 Gerador de Rotas

Com o objetivo de agilizar a realização de testes, foi implementado um gerador de rotas no espaço urbano da cidade de Goiânia. O gerador é configurado com as coordenadas de um retângulo que delimita a área de interesse, que por sua vez contém a cidade de Goiânia:

- Latitude Máxima: -16.6296
- Latitude Mínima: -16.7638
- Longitude Máxima: -49.2194
- Longitude Mínima: -49.3334

O gerador recebe como parâmetros os seguintes valores:

- Quantidade de pontos de entrega
- Coordenadas do centro de distribuição

Considerando os limites da cidade acima definidos, um gerador de números aleatórios é usado para definir pontos de entrega.

Por fim, utilizamos a Open Source Routing Machine[2] para definir uma rota do Centro de Distribuição para cada ponto de entrega. Uma rota é composta por *Steps* (passos), coordenadas nas quais o entregador pode realizar ações que mudem sua trajetória, de acordo com a quantidade de pontos de entrega definida pelo usuário.
O exemplo abaixo ilustra o uso do gerador de rotas para gerar 3 rotas com 1 ponto de entrega intermediário:

generator.generateRoutes(3,1)

As rotas geradas podem ser observadas na Tabela 1.

Tabela 1. Rotas Geradas

Rotas	Centro de Distribuição	Entrega 1	Entrega 2
Rota 1	-49.2576, -16.6800	-49.260251, -16.681666	-49.283726 ,-16.673946
Rota 2	-49.2576, -16.6800	-49.25817, -16.681058	-49.271576, -16.687007
Rota 3	-49.2576, -16.6800	-49.257295, -16.682307	-49.264462, -16.695994

[2] http://project-osrm.org/

4.2 Cenário de Testes

Os resultados obtidos no cenário de testes são demonstrados na prática a partir da utilização das rotas geradas na subseção 4.1.

O cenário é composto por 3 rotas de entrega descritas na Tabela 1. Na Tabela 2, estão demonstrados 2 pontos de coleta gerados e na Tabela 3 são apresentadas 3 restrições definidas no cenário.

Tabela 2. Pontos de Coleta

Coleta1	-49.2833,-16.6871
Coleta2	-49.2848,-16.7142

Tabela 3. Restrições Definidas

Descrição	Peso	Valor Máximo
Distância	2	5000
Tempo	10	100
Desvio	5	12000

O funcionamento das restrições pode ser explicado tomando como exemplo a restrição de distância. Ela possui peso 2 na função de utilidade e teto de 1000 metros, ou seja, caso o ponto de coleta esteja a mais de 1000 metros de distância em relação a qualquer ponto da rota, esta rota automaticamente será considerada inviável para realizar a coleta, assim atribuindo a ele o valor de utilidade 0.0.

Para o cálculo da distância, do tempo e do desvio da rota é utilizado o Open Source Routing Machine, associado ao mapa da cidade de Goiânia utilizando os dados do Open Street Maps[3].

Utilizando a API do OSRM, cada ponto da rota de entrega tem sua distância calculada em relação a cada ponto de coleta. No exemplo da Figura 2. utilizamos o ponto de Coleta 1, sendo então a menor distãncia da Rota 1 em relação à Coleta 1 aquele em relação ao ponto de Entrega 2, com valor calculado de 2300m.

Rota 1	-49.2576, -16.6800	-49.260251, -16.681666	-49.283726 ,-16.673946
Coleta1	-49.2833,-16.6871		
Distância (m)	3300	3000	2300

Figura 2. Cálculo da Distância

O cálculo do tempo é feito de forma similar, apenas alterando o parâmetro, de distância para tempo. Um exemplo é mostrado na Figura 3, que mostra o cálculo do tempo de percurso da Rota 1 em relação à Coleta 1, sendo o menor tempo calculado 180s.

Rota 1	-49.2576, -16.6800	-49.260251, -16.681666	-49.283726 ,-16.673946
Coleta1	-49.2833,-16.6871		
Tempo (s)	270	240	180

Figura 3. Cálculo do Tempo

O desvio é calculado a partir da soma das distâncias saindo de um ponto de entrega, indo até o ponto de coleta e retornando à rota no próximo ponto de entrega. Um exemplo é mostrado na Figura 4, que apresenta os desvios calculados, sendo o menor de 6700 metros saindo do ponto de Entrega 2, indo até o ponto de Coleta 1 e retornando à rota no ponto de Entrega 3.

Rota 1	-49.2576, -16.6800	-49.260251, -16.681666	-49.283726 ,-16.673946
Coleta1	-49.2833,-16.6871		
Desvio (m)	7000	6700	

Figura 4. Cálculo do Desvio

Agrupando todos os menores valores da Rota 1 em relação à Coleta 1, obtemos a Tabela 4.

[3] https://www.openstreetmap.org/#map=14/-16.6952/-49.2454&layers=N

Tabela 4. Rota 1 em relação à Coleta 1

Distância (m)	Tempo (s)	Desvio (m)
2300	180	6700

Cada valor da Tabela 4 é multiplicado pelo peso estabelecido na restrição correspondente e somado aos demais, chegando assim ao valor de utilidade, conforme ilustrado na Figura 5.

Figura 5. Utilidade da Rota 1 em relação à Coleta 1

4.2.1 Normalização dos valores de utilidade

Para normalizar os valores de utilidade para a faixa de 0.0 a 1.0, fazemos uso da função Min-Max definida pela seguinte equação:

$$z = \frac{x - min\,(x)}{max\,(x) - min\,(x)}$$

A tabela com os valores de utilidades de todas as rotas em relação às entregas é percorrida para chegar aos valores mínimos e máximos. Após aplicar a fórmula de normalização, o valor da utilidade é subtraído de 1, para que se tenha 1.0 como utilidade máxima e 0 como utilidade mínima. A Tabela 5 apresenta um exemplo de tabela de utilidades normalizadas.

Tabela 5. Quadro de Utilidades Normalizadas

Coleta 1	Rota 1	1.0
Coleta 1	Rota 2	0.6169809
Coleta 1	Rota 3	0.9646029
Coleta 2	Rota 1	0.0
Coleta 2	Rota 2	0.0
Coleta 2	Rota 3	0.62211215

A partir da Tabela 5, observamos que a Rota 1 é a mais eficiente para realizar a Coleta 1, pois possui o maior valor de utilidade. Da mesma forma, a Rota 3 é a mais eficiente para realizar a Coleta 2. Podemos também observar que as Rotas 1 e 2 possuem utilidade 0.0 para realizar a Coleta 2 devido às suas distâncias apresentarem valores maiores que o valor máximo definido na restrição.

5 Conclusão e Trabalhos Futuros

Um protótipo baseado no cenário de teste foi implementado em Java. Nas próximas etapas, a implementação será refatorada com a definição e implementação da Linguagem de Modelagem Específica de Domínio e do Conhecimento Específico de Domínio.

A implementação do protótipo focou no desenvolvimento fim-a-fim dos mecanismos que farão parte da implementação da linguagem específica de domínio e do conhecimento específico de domínio, a fim de demonstrar a viabilidade da abordagem geral para flexibilizar a criação de aplicações por parte dos especialistas de domínio. Sendo assim, não foi possível definir, no conhecimento específico de domínio, restrições diversas que poderão ser implementadas em trabalhos futuros. Dentre essas restrições, podemos citar a: capacidade de armazenamento do veículo utilizado nas entregas; o uso de recursos de cidades inteligentes para definir rotas mais otimizadas baseadas em dados em tempo real da cidade; pontos de coleta a serem associados à rota do entregador enquanto ele já está realizando o percurso; e definição da quantidade máxima de entregadores que poderão ser utilizados para coletas.

6 Referências Bibliográficas

[1] COSTA, Fábio M. et al. Model-driven domain-specific middleware. In: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS). IEEE, 2017. p. 1961-1971.

[2] Allison, M., Morris, K. A., Costa, F. M., & Clarke, P. J. (2014). Synthesizing interpreted domain-specific models to manage smart microgrids. Journal of Systems and Software, 96, 172-193.

[3] MELO, Paulo César F. Model-Driven Middleware to Support Dynamic Applications for different Domains in Smart Cities. 2018.

[4] CARDENAS, Ivan et al. City logistics, urban goods distribution and last mile delivery and collection. Competition and regulation in network industries, v. 18, n. 1-2, p. 22-43, 2017.]